Gaussian Processes With Categorical Inputs

Sean-Michael Tulloch
MSc by Research in Mathematics of Complex Systems

October 2012

Abstract

Regression problems arise in a variety of contexts including the development of Gaussian process models for computer simulators. Many approaches already exist for Gaussian process regression with continuous valued inputs, however many simulators (and observational data sets) contain both continuous and discrete valued inputs. There are relatively few approaches for addressing Gaussian process regression with mixed continuous and categorical inputs. These include treed Gaussian processes, Dirichlet processes with Generalized Linear Models, and Gaussian processes which use a Hypersphere parameterization.

The aim of this work is to extend Gaussian process models such that they can use categorical inputs e.g. someone's occupation, {Student, Lecturer...}, alongside the usual continuous inputs. A naive approach would be to fit independent Gaussian processes for each category, but this quickly gets inefficient as the number of categories, and in particular the number of categorical inputs, increases. In this work we propose to model the categorical inputs by including a mapping from each categorical element to a continuous real value. We propose to learn the categorical mapping using likelihood based methods. The posterior distribution of the categorical mappings and their relation are expected to reflect their relative influence on the output. Using examples we illustrate the learning dynamics of our method. We explore the strongly multi-modal nature of the posterior distributions for the mappings of the categorical data into real values. We contrast the plug-in estimators which are obtained using likelihood methods with a Bayesian approach using MCMC. Comparisons between our approach and other existing methods for categorical inputs are made on simple data sets.

Acknowledgements

This project would not have been possible without the support of many people. Many thanks to my supervisors, Dan Cornford and Alexis Boukouvalas who have guided me through this project and have carefully read numerous revisions of my thesis. Also thanks to Remi Barillec who helped me understand the FluTE simulator. I would also like thank Alex Brulo, who enabled me to run heavy computational jobs on clusters.

And finally, thanks to my mum, brother, and numerous friends who endured this long process with me, always offering support and guidance where possible.

Contents

List of Figures

Notation And Acronyms

Notation

- $N(x|a,b)$ - is the Normal distribution over the random variable x with mean a and variance b
- $N(\mathbf{x}|\mathbf{a},B)$ - is the multivariate Normal distribution over the random variable $\mathbf{x}$ which has mean $\mathbf{a}$, and co-variance matrix B.
- $\mathbf{x}_i$ - is the i-th data point in the set S with dimensionality 1.
- $\mathbf{x}_i = (x_1,\ x_2\ \ldots\ , x_D)^T$ - i-th data point in the set S with the dimensionality D.
- $\mathbf{x}_i^* = (x_1,\ x_2,\ \ldots\ , x_D)^T$ - is a D-dimensional test set point.
- y_i^* - is the corresponding scalar function output for the test point $\mathbf{x}_i^*$.
- y_i - is the corresponding scalar function output for the point $\mathbf{x}_i$.
- $GP(m(.), k(.\ ,\ .))$ - Gaussian process with a mean function, $m(.)$, and kernel function, $\mathrm{k}(\mathbf{x}_i,\ \mathbf{x}_j)$, $\forall$ pairs of inputs $\mathbf{x}_i$ and $\mathbf{x}_j$.
- $\mathbf{x} \sim p(.)$ - is the random variable $\mathbf{x}$ drawn from probability distribution, $p(.)$.
- X - a set containing the points $(\mathbf{x}_1, ..., \mathbf{x}_n)$ (used for a set of training points).
- X^* - is a set containing d test points.
- $K(X,X)$ - is the kernel between all n training points in the set X, where $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$
- $f(\mathbf{x}_i)$ is the noise-free output, and $\mathrm{y}(\mathbf{x}_i)$ is the noisy-output for the corresponding point $\mathbf{x}_i$. For linear regression models, $f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$.
- $tr(A)$ is the trace of matrix A
- $|A|$ is the determinant of matrix A
- $eig(K(X,X))$ is the matrix of eigenvalues for the co-variance matrix evaluated at the training inputs, X.

Acronyms

- GP - Gaussian Process
- GPs - Gaussian Processes
- MVN - Multivariate Normal distribution
- LML - Log Marginal Likelihood

- ML - Maximum Likelihood
- MAP - Maximize A Posterior
- SCG - Scaled Conjugate Gradient
- HMC - Hybrid Monte Carlo
- SMSE - Standardized Mean Squared Error
- RMSE - Root Mean Squared Error
- NLPD - Negative Likelihood Predictive Density
- MH - Metropolis Hastings
- ST - Simulated Tempering
- SA - Simulated Annealing
- PDUDE - Positive Definite with Unit Diagonal Elements
- TGP - Treed Gaussian Process(es)
- DP - Dirichlet Processes
- DP-GLM - DPs with Generalized Linear Models
- LLM - Limiting Linear Models
- NR - Newton Raphson
- BIC - Bayesian Information Criterion
- AIC - Akaike Information
- MSE - Mean Squared Error

Chapter 1

Introduction

goals of this thesis

In this thesis we will discuss our approach to building a Gaussian Process (GP) model which is able to handle inputs, $\mathbf{x}_i$, which contain a mixture of continuous and categorical elements. These elements are referred to as quantitative and qualitative factors respectively. Two types of qualitative factors, *nominal* and *ordinal* have been considered in defining our GP. This GP model is referred to as the Embedded GP. We focus primarily on the regression task, however Section 6.1 explores using our GP for the classification task.

various existing models for continuous valued inputs only

Consider existing models such as linear regression and GPs, which are used in cases where data inputs $\mathbf{x}_i$ only contains continuous elements.

- Simple linear (in parameters) regression is where the output is expressed as a linear combination of fixed basis functions (which depend on inputs), hence $f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$. It has advantages such as its easy implementation and interpretability. The big drawback with this methodology is that, if we have a complex data set, where the relationship between the variables cannot be approximated by a linear function, we can obtain poor predictions at test points as the model lacks expressive power (Rasmussen 2006, p.8).
- Gaussian Processes (GP) open the possibility of flexible non-parametric models (Rasmussen 2006, p.8). Gaussian processes are often used as a flexible Bayesian prior, where they express beliefs about the underlying function we are modeling (Rasmussen 2006, p.13).

In practice, these models are built using the training set, where the relationship between input and target variables can be learned. Training sets are used to learn the best parameters for the model. Once this is achieved, the model can be used to make predictions on a test point, $\mathbf{x}_i^*$. Test data points are used to assess the fitness of the model and report on measures such as the expected generalization error (Feng 2006).

inclusion of categorical elements

Now, suppose that not all elements of a particular input $\mathbf{x}_i$ (which is D-dimensional) are continuous, but rather there are A quantitative factors, and $D - A$ qualitative factors. Two approaches have been identified in the existing literature to enable GPs to handle inputs such as these.

Existing Approaches

The first approach, described by Broderick & Gramacy (2010) uses the combination of GPs and treed partitioning. Treed GPs take a local-and-divide conquer

approach to non-stationary modeling (Broderick & Gramacy 2010). These models partition the input space using binary splits on a single variable (eg. testScore > 0.9). The second approach, described in Zhou et al. (2010), models the correlation between qualitative factors using Hyper-sphere parameterization. Note that is an extension of the work done by Qian et al. (2008).

Other Existing Approaches

The two methods discussed above are not the only ways of building models which handle data inputs, $\mathbf{x}_i$ with qualitative elements. Qian et al. (2008) consider using restricted correlation functions to model positive correlations between categories whilst the parameterization offered by Zhou et al. (2010) handles both negative and positive correlation between categories.

structure of this thesis

The thesis is structured in the following way. We shall begin by reviewing the existing literature on Gaussian processes (for regression) and related topics, such as sampling, in Chapter 2. Chapter 3 reviews existing models that use data inputs which are a mixture of qualitative and quantitative factors. Then in Chapter 4, we will define the Embedded GP which can handle both quantitative and qualitative inputs. We will map each categorical input to a continuous value (hence real numbers) using embedded mappings. This chapter also compares existing GPs explored in Chapter 3 with the Embedded GP, in terms of analyzing the benefits and disadvantages of using each GP model. This chapter then concludes with providing some useful suggestions on how to use the Embedded GP in practice, along with the most general implementation of this GP. Chapter 5, considers how multi-modality could occur in likelihood and posterior surfaces. This chapter also considers integrating out the uncertainty over hyper-parameters using sampling techniques and crude Monte Carlo on example data sets.

Data sets are derived from a range of sources, which are used for GP modeling. The Embedded GP model is compared against the brute force approach of training independent GPs for each category. These GPs are also compared against other existing methodologies discussed in Chapter 2.6. Validation measures, such as Dawid score are used to evaluate the performance of each of the models. Chapter 5 also reports on the best GP model for each data set based on validation metrics which are calculated using the mean and variance of predictive distributions. These scores consider both resolution and reliability.

Chapter 6 considers extending the Embedded GP model to other tasks such as classification and model selection.

Chapter 2

Review of Gaussian Processes for Continuous Inputs

Goals of this chapter

This section reviews Gaussian processes (GPs) for regression. The marginal likelihood (which marginalizes over the latent functions) and posterior distribution on functions will be discussed. Various sampling routines, such as Simulated Tempering (ST) will be reviewed, along with some useful convergence diagnostics. Validation metrics used to assess probabilistic models shall also be discussed.

2.1 Introduction to Gaussian Processes

Gaussian process co-variance and mean functions

Gaussian processes are collections of random variables, which are used to describe a distribution over functions. They are formally defined as (Rasmussen 2006, p.13):

$$f(\mathbf{x}) \sim GP(m(.), k(.,.)),$$

where,

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{2.1}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))^T (f(\mathbf{x}') - m(\mathbf{x}'))]. \tag{2.2}$$

marginalization property

The prior mean and co-variance over the random function $f(\mathbf{x})$ is given by Equations (2.1) and (2.2) (Rasmussen 2006, p.13). Often in the literature, Equation (2.1) taken to be the zero vector (Rasmussen 2006, p.13). Gaussian processes also have the *consistency* condition (also known as the marginalization property) which implies that if we have the distribution,

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} \sim GP(\boldsymbol{\mu}, \Sigma),$$

where $\mathbf{a} = (f_1, ..., f_c)$ and $\mathbf{b} = (f_{c+1}, ..., f_w)$, then $\mathbf{a} \sim GP(\boldsymbol{\mu}_1, \Sigma_{11})$. Hence the examination of a larger set does not change the distribution of the smaller set (Rasmussen 2006, p.13).

relation to MVN

It is important to say, that everything that applies for Multivariate Normal (MVN)-distributions, still applies to GPs. However a GP, is *not* a MVN distribution because MVNs are defined for a finite set of input points, however GPs take an infinite amount of input points, and are referred to as infinite-dimensional Gaussians (Lawrence 2005).

2.2 Application to Bayesian Linear Models

simple application of a Bayesian linear model

Suppose that functions are drawn from the distribution $f(x) = w_1 x + w_2 x^2$ (which is linear in the parameters), where the prior knowledge on w_i is $w_i \sim N(0, \sigma_i^2)$ and for each pair of w_i and w_j, $\mathbb{E}(w_i w_j) = 0$. Then $f(x)$ has mean, $\mathbb{E}[f(x)] = 0$, and co-variance $\mathbb{E}[f(x)f(x^{'}))] = \sigma_1^2 x x^{'} + \sigma_2^2 x^2 x^{'2}$. We can apply this to other Bayesian linear (in parameter) models.

The function $f(x)$ and prior $p(w)$ induce a Gaussian distribution with mean 0 and co-variance given by $\sigma_1^2 x x^{'} + (\sigma_2^2) x^2 (x^{'})^2$. The function values $f(x_1)$, $f(x_2)$,..., $f(x_n)$ correspond to function values at inputs x_1, ..., x_n, where the f's together make up a joint Gaussian distribution, given the stated prior(s). Hence, n samples can be obtained from the joint distribution,

$$(f(x_1), ..., f(x_n))^T \sim N(\mathbf{0}, K(X, X)),$$

kernel depends on inputs

where K denotes all the element-wise co-variance functions between all pairs of inputs placed together in one matrix and X denotes the set of n training points used to build the GP.

2.3 Making predictions: Noise-free Observations

predict at test points

The interest is not generating random functions from the prior, but rather making predictions about the values $\mathrm{f}(X^*)|\mathrm{f}(X), X, X^*$ which relate to the test inputs X^*. Assuming there are n training points,$\{(\mathbf{x}_i,\ f_i),\ i = 1, ..., n\}$, the joint distribution over $\mathrm{f}(X)$ and $\mathrm{f}(X^*)$ is given by,

$$(\mathrm{f}(X), \mathrm{f}(X^*))^T \sim N\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*) \end{bmatrix}\right),$$

components of kernel

which assumes a zero-mean prior over $\mathbf{f}$. Note that the matrix $K(X, X)$ would be an $n \times n$ matrix (which contains co-variances between the n training points), $K(X, X^*)$ is an $n \times d$ matrix, $K(X^*, X)$ is a $d \times n$ matrix equal to $K(X^*, X)^T$, and $K(X^*, X^*)$ is a $d \times d$ matrix (which contains co-variances between the d test points). The dimensionality of the joint distribution in this case would be $n + d$. To derive the predictive distribution over X^*, requires conditioning on $\mathbf{f}(X)$, that is finding, $\mathrm{f}(X^*)|\mathrm{f}(X), X, X^*$. For brevity, $\mathbf{f} = \mathrm{f}(X)$ and $\mathbf{f}^* = \mathrm{f}(X^*)$.

predictive distribution using a GP prior with zero mean

After some derivations (using arguments from (Rasmussen 2006, p.201), (Bishop 2006, p.93)), the predictive distribution for the noise-free outputs (given noise-free outputs for the training points) is given by:

$$\mathbf{f}(X^*)|\mathbf{f}(X), X, X^* \sim N(\hat{\boldsymbol{\mu}}, \hat{\Sigma}), \tag{2.3}$$

where,

$$\hat{\boldsymbol{\mu}} = K(X^*, X)K(X, X)^{-1}\mathbf{f},$$

$$\hat{\Sigma} = K(X^*, X^*) - K(X^*, X)K(X, X)^{-1}K(X, X^*)).$$

The predictive mean at each of the test points, X^* is a linear combination of all the training point outputs, which is sometimes referred to as a linear predictor

(Rasmussen 2006, p.17). The mean prediction at a single test point could be re-written as:

$$f(\mathbf{x}_i^*) = \sum_{i=1}^{n} A_i f_i,$$

where $\mathrm{A} = K(\mathbf{x}_i^*, X)K(X, X)^{-1}$.

The predictive distribution using GP priors which a have a non-zero mean function is given by

predictive distribution for $GP(m, K)$

$$\mathrm{f}(X^*)|\mathrm{f}(X), X, X^* \sim N(\boldsymbol{\mu}_* + K(X^*, X)K(X, X)^{-1}(\mathbf{f} - \boldsymbol{\mu}), ... \qquad (2.4)$$

$$...K(X^*, X^*) - K(X^*, X)K(X, X)^{-1}K(X, X^*)),$$

where $\boldsymbol{\mu}_*$ is the mean function evaluated at test point $\mathbf{x}_i^*$, and $\boldsymbol{\mu}$ is the mean function evaluated at each of the training inputs, $\mathbf{x}_i$. In this case, the mean function is assumed to be fixed.

2.4 Bayes' Formula and Marginal Likelihood

Bayes' theorem states that,

brief overview Bayes' formula

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)},$$

where $P(A|B)$ is the posterior over the event A, $P(A)$ represents the informative/ uninformative prior over event A, and $P(B|A)$ is the likelihood (Rasmussen 2006, p.200).

Priors can either come from scientific knowledge of the physical process or from previous empirical evidence, however these priors need to be chosen carefully because if inappropriate priors are chosen then we can get wrong judgments.

The difference between uninformative and informative priors is that uninformative priors tend to provide very little information relative to the experiment and have very little influence on the posterior distribution (Bishop 2006), whereas informative priors tend to summarize the evidence from many sources and may have a huge impact on the results. Priors are used incorporate our beliefs about the hyper-parameters in question. $P(B)$ is the marginal likelihood; or alternatively called the normalization constant (Rasmussen 2006, p.18-19) that can alternatively can be expressed as $\int P(B|A)P(A)\, dA$, where this is marginalization over event A (Lauwerrs et al. 2009).

In GP regression, assuming $\mathrm{f}(X) \sim GP(\mathbf{0}, K_s)$, noisy outputs, y_i, $\forall i$ can be generated using $y_i = f(\mathbf{x}_i) + N(0, \sigma^2)$. The normalization term is,

$$p(\mathbf{y}|X) = \int P(\mathbf{y}|\mathbf{f}, X)P(\mathbf{f}|X)\, d\mathbf{f},$$

which is alternatively expressed as,

$$p(\mathbf{y}|X) = \int N(\mathbf{y}|\mathbf{f}, \sigma^2 I)GP(\mathbf{f}|\mathbf{0}, K_s)\, d\mathbf{f}.$$

However by applying properties of MVN-distributions (Bishop 2006) the normalization term becomes,

$$p(\mathbf{y}|X) \sim N(\mathbf{y}|\mathbf{0}, \sigma^2 I + K_s) = N(\mathbf{y}|\mathbf{0}, K). \qquad (2.5)$$

parameters in θ

The kernel matrix K will contain hyper-parameters, such as the length-scale, signal and noise variances, which are contained in vector, $\boldsymbol{\theta}$ (Bishop 2006, p.20). The noise variance is a term which often taken as input-independent (Snelson et al. 2004). Each hyper-parameter stored in the vector $\boldsymbol{\theta}$ is denoted by element $\theta_p, p = 1, ..., a$, where a is the number of hyper-parameters in the GP model.

2.5 Choosing Suitable Values for Hyper-parameters

Marginal Likelihood

Hyper-parameters which appear in the log marginal likelihood need to determined, and is not immediately obvious what values these hyper-parameters should take. However the optimal values are found using the training data, and take the values which maximizes the log marginal likelihood seen below (Bishop 2006, p.112-3),

$$\log\left[p(\mathbf{y}|X,\boldsymbol{\theta})\right] = \frac{1}{2}\mathbf{y}^T K^{-1}\mathbf{y} - \frac{1}{2}\log|K| - \frac{n}{2}\log(2\pi). \tag{2.6}$$

The only term to involve the observed outputs, y_i, over all training data points, is $\frac{1}{2}\mathbf{y}^T K^{-1}\mathbf{y}$, which is the data-fit. The complexity term which involves the training inputs and co-variance function is given by $\frac{1}{2}\log|K|$, whilst term, $\frac{n}{2}\log(2\pi)$ is the normalization constant for marginal likelihood, which is Gaussian (Bishop 2006, p.113). The optimal values chosen for each parameter, $\theta_p, p = 1, ..., a_c$, are chosen such that they satisfy,

$$\frac{\partial}{\partial\theta_p}\log\left[p(\mathbf{y}|X,\boldsymbol{\theta})\right] = \frac{1}{2}\mathbf{y}^T K^{-1}\frac{\partial K}{\partial\theta_p}K^{-1}\mathbf{y} - \frac{1}{2}tr\left(K^{-1}\frac{\partial K}{\partial\theta_p}\right) = 0, \tag{2.7}$$

where a_c represents the number of hyper-parameters which appear in the co-variance kernel, K. Finding solutions which satisfy Equation 2.7 will lead to a point-estimation of the parameters θ_p. Instead of maximizing the log marginal likelihood, the log posterior distribution could be maximized over $\boldsymbol{\theta}$,

Maximize the Posterior

$$\log\left[p(\boldsymbol{\theta}|X,\mathbf{y})\right] = \frac{1}{2}\mathbf{y}^T K^{-1}\mathbf{y} - \frac{1}{2}\log|K| - \frac{n}{2}\log(2\pi) + \log\left[p(\boldsymbol{\theta})\right] - \log\left[p(\mathbf{y}|X)\right]. \tag{2.8}$$

The difference between the two approaches is that Equation 2.8 has an additional term which incorporates prior knowledge about $\boldsymbol{\theta}$ into the estimation procedure. The log prior, $\log\left[p(\boldsymbol{\theta})\right]$, is the prior over the hyper-parameters in the GP prior. Once the optimal solutions $\hat{\boldsymbol{\theta}}$ have been found either by performing MAP or ML, they can be substituted into $p(y_i^*|\mathbf{x}_i^*, X, \mathbf{y}, \hat{\boldsymbol{\theta}})$. Performing MAP or ML could lead to finding multiple local maxima that exist in the log marginal/posterior distribution, where each mode refers to a particular interpretation of the data (Bishop 2006, p.115). This occurs because the model cannot confidently reject the possible multiple possibilities.

MAP: GP with non-zero mean function

So far the presentation assumes that the GP prior has a non-zero mean. MAP and ML estimation can also be applied to a GP prior with a non-zero mean function. Instead of maximizing Equation 2.8 for MAP estimation, one would maximize,

$$\log\left[p(\boldsymbol{\theta}|X,\mathbf{y})\right] = \frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^T K^{-1}(\mathbf{y}-\boldsymbol{\mu}) - \frac{1}{2}\log|K| - \frac{n}{2}\log(2\pi) + \log\left[p(\boldsymbol{\theta})\right] - \log\left[p(\mathbf{y}|X)\right], \tag{2.9}$$

where $\boldsymbol{\mu}$ is a known $n \times 1$ vector corresponding to the mean function $m(.)$ being evaluated at each of the training points. Hyper-parameters, $\boldsymbol{\theta}$ can enter into the mean or kernel functions. To optimize the parameters, $\theta_p, p = 1, ..., a_c$, in the co-variance function and $\theta_s, s = 1, ..., a_m$, in the mean function one needs to use the

equations,

$$\frac{\partial}{\partial\theta_p}\log\left[p(\boldsymbol{\theta}|X,\mathbf{y})\right] = \frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^T K^{-1}\frac{\partial K}{\partial\theta_p}K^{-1}(\mathbf{y}-\boldsymbol{\mu}) - \frac{1}{2}tr\left(K^{-1}\frac{\partial K}{\partial\theta_p}\right) + \log(\frac{\partial\log\left[p(\boldsymbol{\theta})\right]}{\partial\theta_p}) = 0, \tag{2.10}$$

$$\frac{\partial}{\partial\theta_s}\log\left[p(\boldsymbol{\theta}|X,\mathbf{y})\right] = -(\mathbf{y}-\boldsymbol{\mu})^T K^{-1}(\frac{\partial m}{\partial\theta_s}) + \log(\frac{\partial\log\left[p(\boldsymbol{\theta})\right]}{\partial\theta_s}) = 0, \tag{2.11}$$

where $\frac{\partial m}{\partial\theta_s}$ represents partially differentiating the mean function $m(.)$ with respect to hyper-parameter, θ_s.

Using basis functions in GP prior mean function

Certain types of mean function $m(\mathbf{x}_i)$ may be written as a linear combination of r_b basis functions, h_j,

$$m(\mathbf{x}_i) = h_1(\mathbf{x}_i)\beta_1 + h_2(\mathbf{x}_i)\beta_2 + ... + h_{r_b}(\mathbf{x}_i)\beta_{r_b} = h(\mathbf{x}_i)^T\boldsymbol{\beta}, \tag{2.12}$$

where $j = 1, ..., r_b$. This particular form for the mean function used in the GP prior leads to

$$f(\mathbf{x}) \sim GP(h(\mathbf{x})^T\boldsymbol{\beta}, k(\mathbf{x},\mathbf{x}^{'})). \tag{2.13}$$

The regression parameters, $\beta_i, i = 1, ..., r_b$, can be analytically integrated out using a Multivariate Normal (MVN) prior, $MVN(\beta|\mathbf{b}, B)$, although parameters in the kernel, K cannot be integrated out analytically, which is why sampling routines are used to sample from the posterior distribution, $P(\boldsymbol{\theta}|X,\mathbf{y})$, over the remaining parameters, $\boldsymbol{\theta}$. Terms $\mathbf{b}$ and B remain fixed. Integrating $\boldsymbol{\beta}$ analytically leads to,

$$\int GP(h(\mathbf{x})^T\boldsymbol{\beta}, k(\mathbf{x},\mathbf{x}^{'}))MVN(\boldsymbol{\beta}|\mathbf{b}, B)\, d\boldsymbol{\beta} = GP(h(\mathbf{x})^T\mathbf{b}, k(\mathbf{x},\mathbf{x}^{'}) + h(\mathbf{x})^T B h(\mathbf{x}^{'})) \tag{2.14}$$

The log of the posterior distribution, $p(\boldsymbol{\theta}|D)$ is given by

$$R = \log(p(\mathbf{y}|X,\mathbf{b},B,\boldsymbol{\theta})) + \log(p(\boldsymbol{\theta})) - \log(p(\mathbf{y}|X)) = ...$$

$$... - \frac{1}{2}(\mathbf{y} - (\mathbf{h}(X))^T\mathbf{b})^T(K + (\mathbf{h}(X)^T)B(\mathbf{h}(X)))^{-1}(\mathbf{y} - \mathbf{h}(X)^T\mathbf{b}) + ...$$

$$... - \frac{1}{2}\log|(K + (\mathbf{h}(X))^T)B(\mathbf{h}(X))| - \frac{n}{2}\log(2\pi) + \log(p(\boldsymbol{\theta})) - \log(p(\mathbf{y}|X)),$$

which also includes the prior $p(\boldsymbol{\theta})$. Maximization of R instead of Equation 2.9 can be used to find the optimal values, $\hat{\boldsymbol{\theta}}$. Predictions at test points, $\mathbf{x}_i^*$ are made by plugging the mean and variance function from the GP prior seen in Equation 2.14, in the predictive distribution

$$\mathbf{y}(X^*)|\mathbf{y}(X), X, X^* \sim N(\boldsymbol{\mu}_* + K(X^*,X)K(X,X)^{-1}(\mathbf{y}-\boldsymbol{\mu}), ... \tag{2.15}$$

$$...K(X^*,X^*) - K(X^*,X)K(X,X)^{-1}K(X,X^*)),$$

where this assumes the mean function is fixed. Note that $\boldsymbol{\mu}_*$ represents the mean function evaluated at each of the test point in the set X^*. However, there is a third option, which is to integrate out the uncertainty over the remaining hyper-parameters, $\boldsymbol{\theta}$.

2.6 Sampling Techniques and Convergence Diagnostics

Optimal values for $\boldsymbol{\theta}$ may be found by maximizing the posterior distribution, $p(\boldsymbol{\theta}|X,\mathbf{y})$, where these entered into the predictive distribution, $p(y_i^*|\mathbf{x}_i^*, X, \mathbf{y}, \hat{\boldsymbol{\theta}})$ as plug-in values (Walsh 2004). Instead of using these plug-in estimates, another option

would be to integrate out the uncertainty over these parameters (Taddy et al. 2010), where the predictive distribution would be,

$$p(y_i^*|\mathbf{x}_i^*, X, \mathbf{y}) = \int p(y_i^*|\mathbf{x}_i^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|X, \mathbf{y}) \; d\boldsymbol{\theta}. \tag{2.16}$$

Calculating this integral analytically is not possible in most circumstances, for example hyper-parameters, such as the length scales which appear in the kernel, K of the GP prior could not be integrated out. To approximate this integral, sampling routines such as Metropolis Hastings (MH) can be applied where these attempt to simulate samples the posterior distribution, $p(\boldsymbol{\theta}|X, \mathbf{y})$. Once samples are obtained from posterior distribution, $p(\boldsymbol{\theta}|X, \mathbf{y})$, these samples are used to approximate the integral as shown in Equation 2.16, which can be expressed as an expectation over the sampled distribution, $p(\boldsymbol{\theta}|X, \mathbf{y})$ such that,

$$p(y_i^*|\mathbf{x}_i^*, X, \mathbf{y}) \simeq \frac{1}{F} \sum_{c=1}^{F} p(y_i^*|\mathbf{x}_i^*, \boldsymbol{\theta}^{(c)}) \simeq E_{p(\boldsymbol{\theta}|X,\mathbf{y})} p(y_i^*|\mathbf{x}_i^*, \boldsymbol{\theta}),$$

where this is referred to as Monte Carlo integration (Titsias et al. 2009).

Some well-known sampling routines such as MH and Simulated Tempering (ST) which can be used to sample $p(\boldsymbol{\theta}|X, \mathbf{y})$ are reviewed next.

2.6.1 Metropolis-Hasting Sampling

Suppose that the (arbitrary) distribution, $p(\boldsymbol{\theta})$, can expressed as $\frac{f(\boldsymbol{\theta})}{Z}$, where $Z = \int f(\boldsymbol{\theta}) \; d\boldsymbol{\theta}$, is the normalization constant for this distribution. A benefit of using MH, is that the normalization constant, Z is not required. Proposal distributions, which are also referred to as candidate-generating or proposal distributions, $q(\boldsymbol{\theta}^{(i)}|\boldsymbol{\theta}^{(i-1)})$ are used in MH to generate the sequence of values $\boldsymbol{\theta}^{(i)}$, at each iteration i. If $p(\boldsymbol{\theta})$ is a multi-dimensional distribution, there are two forms of the MH algorithm in the literature, the component-wise and block-wise updating (Johnson et al. 2011). Block-wise MH requires that the proposal distribution, $q(\boldsymbol{\theta}^{(i)}|\boldsymbol{\theta}^{(i-1)})$ has the same dimensionality has the distribution, $p(\boldsymbol{\theta})$. If $p(\boldsymbol{\theta})$ involves N variables, so will $q(\boldsymbol{\theta}^{(i)}|\boldsymbol{\theta}^{(i-1)})$. The notation, $\boldsymbol{\theta}^{(i)} = (\theta_1^{(i)}, ..., \theta_N^{(i)})$ represents the i-th state of block-wise MH sampler. Below are the steps to sample from distribution, $p(\boldsymbol{\theta})$, using block-wise MH,

block-wise Metropolis-Hastings algorithm

1. Start at iteration, i=1.
2. Use an initial starting position for each $\theta_j^{(1)} \; \forall j$, in the vector, $\boldsymbol{\theta}^{(1)}$.
3. Generate a sample using the proposal density, $\boldsymbol{\theta}^{(test)} \sim q(\boldsymbol{\theta}^{(i+1)}|\boldsymbol{\theta}^{(i)})$.
4. Draw a uniform random number, $U \in [0, 1]$.
5. Evaluate the acceptance probability, α for which $\alpha = \min\left(1, \frac{p(\boldsymbol{\theta}^{(i)})}{p(\boldsymbol{\theta}^{(test)})} \frac{q(\boldsymbol{\theta}^{(test)}|\boldsymbol{\theta}^{(i)})}{q(\boldsymbol{\theta}^{(i)}|\boldsymbol{\theta}^{(test)})}\right)$.
6. If $U \leq \alpha$, then accept the new state, setting $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(test)}$, or otherwise $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)}$ (rejecting the proposed sample).
7. Continue steps 3-6 for each time step i, until $i = i_c$, where i_c is the length of the sequence of draws.

When calculating α, the distribution, $p(\boldsymbol{\theta})$ does not need to used, but rather, $f(\boldsymbol{\theta})$, because the normalization constant which appears in $p(\boldsymbol{\theta})$ cancels out, hence,

$$\frac{p(\boldsymbol{\theta}^{(test)})}{p(\boldsymbol{\theta}^{(i)})} = \frac{f(\boldsymbol{\theta}^{(test)})}{K}\frac{K}{f(\boldsymbol{\theta}^{(i)})} = \frac{f(\boldsymbol{\theta}^{(test)})}{f(\boldsymbol{\theta}^{(i)})}.$$

The problem with block-wise MH is that it may be hard to find suitable proposal distributions, and this algorithm is associated with high rejection rates (Johnson et al. 2011). Another option would be to update each component one at a time, where this is called component-wise MH, which involves proposal distributions involving just one variable. The component, θ_j in $\boldsymbol{\theta}$ which would be updated whilst the other components are fixed. It may be computationally simpler to make proposals such as this. Steps in the component-wise MH algorithm are conditional on each other, hence the updated variables must be used, when fixing each component apart from the one which is to be updated. Note, that the components are updated in order, updating the first component of $\boldsymbol{\theta}$ before moving onto the second component. To generate samples from $p(\boldsymbol{\theta})$ using the component-wise MH algorithm, the following steps must be taken,

component-wise Metropolis Hastings

1. Start at iteration, i=1.
2. Use an initial starting position for each $\theta_j^{(1)}$ in the vector, $\boldsymbol{\theta}^{(1)}$.
3. Start at o =1.
4. Pick the o-th component of $\boldsymbol{\theta}^{(i)}$.
5. Generate a sample using the proposal density, $\theta^{(test)} \sim q(\theta_o^{(i+1)}|\theta_o^{(i)})$.
6. Draw a uniform random number, $U \in [0, 1]$.
7. Evaluate the acceptance probability, α for which $\alpha = \min\left(1, \frac{p(\theta_o^{(i)}, \boldsymbol{\theta}_{\sim o}^{(i)})}{p(\theta^{(test)}, \boldsymbol{\theta}_{\sim o}^{(i)})} \frac{q(\theta^{(test)}|\theta_o^{(i)})}{q(\theta_o^{(i)}|\theta^{(test)})}\right)$.
8. If $U \leq \alpha$, then accept the new state, setting $\theta_o^{(i+1)} = \theta^{(test)}$, or otherwise $\theta_o^{(i+1)} = \theta_o^{(i)}$.
9. Continue steps 4-8 if there is still another component to update at the current time step i. If not, move on to the next time step.
10. Continue steps 3-8 for each iteration i, until $i = i_c$, where i_c is the total number of iterations.

In the above algorithm, $\boldsymbol{\theta}_{\sim o}^{(i)}$ represents all components of $\boldsymbol{\theta}$ apart from the o-th component.

The choice of which two variants of MH to use is commonly unclear (Johnson et al. 2011).

However, MH-methods are very sensitive to the step size of the transition steps, where if these are too small, the sampling routine will become easily trapped in a deep local minimum where it will not be able to escape in practical simulation time. This is a serious issue of slow mixing and if too large, the acceptance rate tends to be very low, where it will tend to ignore some 'local details' about the distribution we are trying to sample from. There are no general guidelines on how to select the appropriate step size of the transition steps (Li & Protopopescu 2004).

2.6.2 Burn-In Period and Mixing

burn-in period

These sampling routines require that a subset of the generated sequence, $\boldsymbol{\theta}^{(i)}$ is disposed of because these particular samples are not samples from the required distribution, $p(\boldsymbol{\theta})$ (Gelman & Shirley 2010). These particular samples are removed as burn-in, during a sufficient burn-in period (which is say of i_d iterations). Typical burn-in periods involve rejecting the first 1000 to 5000 samples in a chain, although it could be longer. Poor choices of starting values for $\boldsymbol{\theta}^{(1)}$ and proposal distribution(s), can also significantly increase the burn-in period (Walsh 2004). Retained samples, $\boldsymbol{\theta}^{(p+1)}, ..., \boldsymbol{\theta}^{(i_c)}$ should be samples from $p(\boldsymbol{\theta})$.

poor/good mixing

Generated chains are either said to be poorly or well mixed. A poorly mixed chain means that the chain will stay in small regions of the parameter space (Gelman & Shirley 2010). This happens because either the hyper-parameters are highly correlated with each other and additionally with multi-modal distributions, the choice of starting values can trap the chain near one of these modes. By checking the acceptance rate of a new sample, $\boldsymbol{\theta}^{(i)}$, this can be a useful diagnostic to observe whether there is poor mixing in the model (Walsh 2004). Two possible approaches have been suggested for sampling multi-modal target distributions, $p(\boldsymbol{\theta})$, where either different chains can be started using different highly dispersed initial values for hyper-parameters or using sampling routines such as Simulated Tempering (ST) (Walsh 2004).

2.6.3 Simulated Schemes: Simulated Annealing and Simulated Tempering

Simulated Annealing

A problem with using a MH algorithm is that it depends on the sensitivity to the step size of the transition steps. It is much easier for a system to escape from a local minimum, by using optimization routines such as Simulated Annealing (SA), which was built for the task of handling non-linear optimization problems. Annealing is the process of heating a solid that permits many atomic arrangements, and then cooling it down in a very slow manner until the material freezes into good crystal, where this analogy is to used to find the global minimum of a (multivariate) function, f (Bertsimas & Tsitsiklis 1993). SA involves setting the initial temperature to a very high value, and then cooling the temperature down very slowly. There are different ways of cooling down the temperature in SA algorithms (Perrin et al. 2005), which include geometric,

$$T_t = T_1 A^t,$$

and logarithmic schemes,

$$T_t = \frac{C}{\log(1+t)},$$

simulated annealing algorithm

where T_1 and T_t are initial and current temperatures at times 1 and t respectively, and variables, A and C are tuning constants. Here, are the steps for the SA algorithm where the goal is to minimize function, $f(\boldsymbol{\theta})$,

1. Select the starting temperature, T_1, and the initial parameters $\boldsymbol{\theta}^{(1)}$, where i=1 and the system has energy value, E_1.
2. Randomly select a new candidate, $\boldsymbol{\theta}^{(p)}$, by perturbing the previous position, $\boldsymbol{\theta}^{(i)}$. The energy at the candidate state, will be E_p.
3. Accept the candidate position $\boldsymbol{\theta}^{(p)}$, based on MH-criterion, hence,
   - if the energy differences, $\triangle E = E_{i+1} - E_i < 0$, accept the new candidate position, hence $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(p)}$

- or otherwise draw a uniform random number, $U \in [0, 1]$ and accept candidate with probability, $p = \exp^{\frac{-\triangle E}{T_{i+1}}}$.

4. Decrease the temperature, with chosen cooling scheme, and repeat steps 2-3, until T close to freezing ($\simeq 0$)

5. Obtain at $T \simeq 0$, the optimal solution $\boldsymbol{\theta}$ which minimizes the function f.

problems with SA

The system evolves according to a MH-criterion (Bertsimas & Tsitsiklis 1993); the temperature is reduced very slowly, using the chosen cooling scheme (Li & Protopopescu 2004). The problem with SA is that the temperature needs to be cooled very slowly or as a result the algorithm will get trapped in another local minimum (Li & Protopopescu 2004). This is likely to happen for multi-dimensional functions (Li & Protopopescu 2004), however it depends on the structure of the function in question. SA is sensitive to the choice of the initial temperature, along with how slowly the temperature is cooled down. A solution to the drawbacks offered by both SA and MH, is to use the sampling routine, Simulated Tempering (ST), which is described by Li & Protopopescu (2004).

Simulated Tempering

The idea behind ST is to introduce an additional dynamical variable, T, which is used in the process of sampling, along with the original variables, θ_j, $j = 1, ..., a$. This allows the dynamical variable, T, to vary on a discrete set of m temperature states, $T_i, i = 1, ..., m$ where $T_1 = 1$. The set of temperatures, T_i vary on a temperature ladder. At each temperature level, T_i, a stationary distribution $\pi_i(\boldsymbol{\theta}\ , T_i)$ is constructed, where $\pi_1(\boldsymbol{\theta}, T_1)$ is the distribution where samples are required. These distributions are also called progressive flat distributions (Behrens et al. 2010). Stationary distributions used in ST are of the form,

$$\pi_i(\boldsymbol{\theta}, T_i) = \exp^{\frac{\log p(\boldsymbol{\theta})}{T_i}}.$$

Because ST is able to move up and down the temperature ladder, this allows the algorithm to escape local minima, and increases its chances of locating multiple minima in the surface, according to the MH-algorithm rule(s). This is advantage of using ST over SA. However the trade-off here is that the ST algorithm needs to spend more time at higher temperature levels, and therefore will generate samples from $\pi_1(\boldsymbol{\theta}, T_1)$ much more slowly, than using the MH sampling routines (Li & Protopopescu 2004).

simulated tempering algorithm

ST requires normalization constants, Z_i for each stationary distribution on the temperature ladder. Li & Protopopescu (2004) suggests using importance sampling, which will obtain fast, but yet good approximations to these constants. The normalization constant, Z_i, at temperature T_i is,

$$Z_i = \int \pi_i(\boldsymbol{\theta}, T_i)\ d\boldsymbol{\theta}.$$

However drawing samples from these unnormalized distributions, $\pi_i(\boldsymbol{\theta}, T_i)$, is not an easy task. To deal with this, we introduce another distribution $G(\boldsymbol{\theta}, T_i)$, which is called the sampling distribution. This distribution is similar to $\pi_i(\boldsymbol{\theta}, T_i)$. Normalization constants can be then calculated as,

$$Z_i = \int \frac{\pi_i(\boldsymbol{\theta}, T_i) G(\boldsymbol{\theta}, T_i)}{G(\boldsymbol{\theta}, T_i)}\ d\boldsymbol{\theta},$$

where c samples, $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, ..., \boldsymbol{\theta}^{(c)}$ (discarding burn-in) are obtained from sampling the distribution, $G(\boldsymbol{\theta}, T_i)$. These samples are used to approximate the normalization constants, Z_i, such that,

$$Z_i \simeq \frac{1}{c} \sum_{j=1}^{c} \frac{\pi_i(\boldsymbol{\theta}^{(j)}, T_i)}{G(\boldsymbol{\theta}^{(j)}, T_i)}.$$

Li & Protopopescu (2004) suggests using a Gaussian distribution for G. The ST algorithm is outlined below,

1. Choose the temperatures T_i on the temperature ladder of size m, and obtain the normalization constants, Z_i for $i = 1, ..., m$. The m temperature levels are indicated by $i = 1, ..., m$.
2. Begin the ST algorithm at the highest temperature in the temperature ladder. Initialize values for $\boldsymbol{\theta}$.
3. At the current temperature, update $\boldsymbol{\theta}$ by using a MH-update which requires distribution, $\pi_i(\boldsymbol{\theta}, T_i)$.
4. Propose increasing or decreasing the temperature, T_i by one temperature level, hence $i \pm 1 = j$, according to temperature transition probabilities, $p_{1,2} = 1.0$ if $i = 1$, $p_{m,m-1}$ if $i = m$ and $p_{i,i+1} = p_{i,i-1} = 0.5$ if $1 < i < m$.
5. A MH-step is used to decide whether to accept or reject the proposed temperature, T_j, based on,
$$\alpha = \frac{Z_j \pi_j(\boldsymbol{\theta}, T_j) p_{j,i}}{Z_i \pi_i(\boldsymbol{\theta}, T_i) p_{i,j}},$$
where the temperature change from i-th to j-th positions in the temperature ladder is given with probability $\min(1, \alpha)$.
6. Repeat steps 3-5 for M number of temperature transitions.
7. Keep the samples which correspond to the lowest temperature $T_1 = 1$.

2.6.4 Convergence Diagnostics

Convergence diagnostics are used to inspect whether the chain from a MCMC algorithm, has converged to the stationary distribution (Gelman & Shirley 2010). Generating samples using sampling routines such as MH, could result in producing poorly mixed chains, where this slows down the convergence of the sampling routine used (Christen & Fox 2005).

Autocorrelation

Adjacent samples from sampling routines such as MH could have positive correlation, where this arises because each sample $\theta^{(i)}$ depends on the previous state, $\theta^{(i-1)}$. This nature of this correlation can be quantified by using the autocorrelation function (Gelman & Shirley 2010). Suppose the chosen sampling routine, generates a sequence of length i_c, correlation can occur between two adjacent members or more generally between more distinct members in the sequence. The autocorrelation function between members, $\theta^{(i)}$ and $\theta^{(i+k)}$, which have a time lag k, is given by,

$$p_k = \frac{\sum_{i=1}^{i_c-k} (\theta^{(i)} - m)(\theta^{(i+k)} - m)}{\sum_{i=1}^{i_c} (\theta^{(i)} - m)^2},$$

where the mean of the samples is given by,

$$m = \frac{1}{i_c} \sum_{i=1}^{i_c} \theta^{(i)}.$$

An important result from time series analysis is that if samples $\theta^{(i)}$ are generated from a stationary yet correlated process, correlated samples may still provide unbiasedness of the target distribution (Gelman & Shirley 2010). One possible way to reduce the autocorrelation between samples is to use thinning, where this stores every m-th point after burn-in of the sequence generated by the sampling routine. The other samples, along with the burn-in samples are discarded (Christen & Fox 2005), therefore reducing the number of samples used for inference. Plotting the autocorrelation against the time lag k should show a geometric decay. Slow decaying correlations in this plot, indicate poor mixing of the chain.

trace plots

One way of demonstrating convergence of sequences from sampling routines is to look at the time series trace, where this plots the random variable, $\theta_j^{(i)}$, against the number of iterations, i. These plots indicate whether a bigger burn-in period should be taken because of the lack of convergence of the sampler. They also indicate whether the chain has been poorly mixed (Gelman & Shirley 2010).

Batch means approach

The batch-means approach can also be used to check the convergence of Markov chains. Batches are subsequences of consecutive iteratives, $\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\theta}^{(k+2)}, ..., \boldsymbol{\theta}^{(k+b)}$, of a Markov chain, where b is the batch length. Assuming the Markov chain is stationary, all batches will have the same joint distribution where the Central Limit Theorem (CLT) is applied to each batch. Given a function, g, batch means are calculated as,

$$\mu_m(b) = \frac{1}{b}\sum_{i=1}^{b} g(\boldsymbol{\theta}^{(k+i)}), \tag{2.17}$$

for batch m, where Equation 2.17 is an Monte-Carlo approximation to the integral, $\int g(\boldsymbol{\theta})p(\boldsymbol{\theta})\, d\boldsymbol{\theta}$, where samples are drawn from distribution, $\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\theta}^{(k+2)}, ..., \boldsymbol{\theta}^{(k+b)} \sim p(\boldsymbol{\theta})$. To access convergence, let the batch length be of a fixed size, b, and divide the chain into m separate batches (after discarding samples from burn-in). For each batch, calculate the batch means estimate, given in Equation 2.17. Each batch, gives batch mean estimates of, $\mu_1(\frac{b}{m})$,...,$\mu_m(\frac{b}{m})$. The differences between two consecutive batch mean estimates, $\mu_i(\frac{b}{m})$ and $\mu_{i+1}(\frac{b}{m})$ should satisfy,

$$\left|\mu_i\left(\frac{b}{m}\right) - \mu_{i+1}\left(\frac{b}{m}\right)\right| \leq \epsilon, \tag{2.18}$$

where the tolerance $\epsilon > 0$ and fixed. Satisfying this condition indicates convergence (Cowles et al. 1999).

Gelman-Rubin diagnostic convergence

another approach to initial conditions

Another way to check for convergence is to use the criteria imposed by Brooks & Gelman (1998). This criteria requires multiple chains to be run, where each chain starts at different initial values for $\boldsymbol{\theta}$. Brooks & Gelman (1998) generated a single-mode algorithm, to locate the high-density regions, and then sampling from a mixture of t-distributions located at these modes to generate suitable starting values. Other ways to locate high-density regions in the distribution surface is to use a non-linear optimization routine such as the scaled conjugate routine algorithm. Having obtained suitable starting conditions, each chain is run for a length of $2k$ where the first k samples are discarded due to burn-in. The retained samples are now said to have reached the stationary distribution. For each chain (after burn-in), the sample mean and variances of $\theta_j^{(i)}$ can be calculated, however with m parallel chains, inferences such as these should be close enough for each parallel chain. Brooks & Gelman

(1998) suggested comparing these to the inference made by mixing the mk samples from all chains. To do this, the within- and between-chain variances need to be worked out. The within-chain variance, W, is given by,

Univariate Potential Scale Reduction Factor

$$W = \frac{1}{m(k-1)} \sum_{p=1}^{m} s_p^2,$$

where $s_p^2 = \sum_{i=1}^{k} (\theta_{j,p}^{(i)} - r_p)^2$ is the variance for the p-th chain and $r_p = \frac{1}{k} \sum_{i=1}^{k} \theta_{j,p}^{(i)}$ and $\theta_{j,p}^{(i)}$ represents a sample for the j-th co-variate in chain p at time state i. The within-chain variance, W is the mean of the variances across all chains. The between-chain variance $\frac{B}{k}$ is,

$$\frac{B}{k} = \frac{1}{(m-1)} \sum_{p=1}^{m} (r_m - r)^2,$$

where $r = \frac{1}{m} \sum_{p=1}^{m} r_p$ represents the mean over the means of samples from each individual chain. The between-chain and within-chain variances are used to approximate the variance of stationary distribution as,

$$Var(\theta_j) = \frac{k-1}{k} W + \frac{m+1}{m} \frac{1}{k} B. \tag{2.19}$$

If the initial values were over-dispersed, then this measure will overestimate the variance of the true stationary distribution, however this measure would be unbiased if the initial values of all chains were drawn from the target distribution. When all chains have reached the target distribution, the variance approximation seen in Equation 2.19 should be very close to the within-chain variance, W. Hence, the ratio of these two measures, $Var(\theta_j)$ and W should be close to one, where the square root of this measure is called the potential scale reduction factor (PSRF) (Brooks & Gelman 1998), which is given by $\frac{Var(\theta_j)}{W}$. If this occurs then the conclusion to be made is that each of the chains have stabilized, and they are likely to have converged to the stationary distribution. This is known as the univariate PSRF. A weakness for this measure is that it is only applicable to one co-variate (of $\boldsymbol{\theta}$) at a time (Venna. 2007). The Multivariate Potential Scale Reduction Factor (MPSRF) considers all components of $\boldsymbol{\theta}$, whilst the univariate PSRFs considers one component of $\boldsymbol{\theta}$ at one time. Univariate PSRFs (calculated for each component of $\boldsymbol{\theta}$) are bounded by the MPSRF. MPSRF uses the within-chain and between-chain variances, $\frac{\boldsymbol{B}}{k}$ and $\boldsymbol{W}$, given by,

Multi-variant Potential Scale Reduction Factor

$$\boldsymbol{W} = \frac{1}{m(k-1)} \sum_{p=1}^{m} \sum_{i=1}^{k} (\boldsymbol{\theta}_p^{(i)} - \boldsymbol{r}_p)(\boldsymbol{\theta}_p^{(i)} - \boldsymbol{r}_p)^T,$$

$$\frac{\boldsymbol{B}}{k} = \frac{1}{m-1} \sum_{p=1}^{m} (\boldsymbol{r}_m - \boldsymbol{r})(\boldsymbol{r}_m - \boldsymbol{r})^T,$$

to calculate the posterior variance-covariance matrix,

$$\mathbf{V} = \frac{k-1}{k} \mathbf{W} + \frac{m+1}{m} \frac{1}{k} \mathbf{B}$$

where $\boldsymbol{r}_p = \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{\theta}_p^{(i)}$ and $\boldsymbol{\theta}_p^{(i)}$ represents the i-th sample from chain p, and $\boldsymbol{r} = \frac{1}{m} \sum_{p=1}^{m} \boldsymbol{r}_p$ (Venna. 2007).

Both measures, PSRF and MPSRF should be as close to 1 as possible, but indication of convergence when both diagnostics are less than 1.2 (Venna. 2007). The maximum univariate PSRF is bounded by the MPSRF (Brooks & Gelman 1998). Both diagnostics will be considered when sampling from the posterior distribution, $p(\boldsymbol{\theta}|X, \mathbf{y})$, where the results will be shown in Chapter 5.

2.6.5 Calculating the Predictive Distribution Using Samples

Different types of sampling routine and convergence diagnostics have been reviewed in this chapter and will be used in Chapter 5. Sampling is used to evaluate expressions such as

$$p(y^*|X^*, D) = \int p(y^*|X^*, D, \boldsymbol{\theta}) p(\boldsymbol{\theta}|X, \mathbf{y}) \; d\boldsymbol{\theta}, \tag{2.20}$$

by generating i_c samples from, $p(\boldsymbol{\theta}|X, \mathbf{y})$, using a sampling routine discussed above. These samples are used to approximate the integral in Equation 2.20 by,

$$p(y^*|X^*, D) \simeq \frac{1}{n} \sum_{i=1}^{n} p(y^*|X^*, D, \boldsymbol{\theta}^{(i)}),$$

where,

$$p(y^*|X^*, D, \boldsymbol{\theta}^{(i)}) = N(E(y_b^*|\mathbf{x}_b^*, D, \boldsymbol{\theta}^{(i)}), Var(y_b^*|\mathbf{x}_b^*, \boldsymbol{\theta}^{(i)}, D))$$

predictive moments

To calculate the mean, variance and co-variance of the distribution, $p(y^*|x^*, D)$, at test points $\mathbf{x}_b^*$ and $\mathbf{x}_a^*$,

$$E(y_b^*|\mathbf{x}_b^*, D) \simeq \frac{1}{i_c} \sum_{i=1}^{i_c} E(y_b^*|\mathbf{x}_b^*, D, \boldsymbol{\theta}^{(i)}), \tag{2.21}$$

$$Var(y_b^*|\mathbf{x}_b^*, D) \simeq \frac{1}{i_c} \left(\sum_{i=1}^{i_c} Var(y_b^*|\mathbf{x}_b^*, \boldsymbol{\theta}^{(i)}, D) \right) + Var(E(y_b^*|\mathbf{x}_b^*, \boldsymbol{\theta}^{(i)}, D)), \tag{2.22}$$

$$Cov((y_b^*|\mathbf{x}_b^*, D), (y_a^*|\mathbf{x}_a^*, D)) \simeq \left(\frac{1}{i_c} \sum_{i=1}^{i_c} Cov((y_b^*|\mathbf{x}_b^*, \boldsymbol{\theta}^{(i)}, D), (y_a^*|\mathbf{x}_a^*, \boldsymbol{\theta}^{(i)}, D)) \right) ...$$

$$... + Cov(E(y_b^*|\mathbf{x}_b^*, \boldsymbol{\theta}^{(i)}, D), E(y_a^*|\mathbf{x}_a^*, \boldsymbol{\theta}^{(i)}, D)), \tag{2.23}$$

where Equation 2.22 uses the law of total variance, and a generalization is seen in Equation 2.23 which is the law of the total co-variance. These formulas are used to calculate validation metrics such as the NLPD and Dawid score.

2.7 Validation

Given training inputs, X and outputs $\mathbf{y}$, GP models learn suitable values for hyper-parameters, $\boldsymbol{\theta}$, by maximizing the posterior distribution (or likelihood) which is based on the evidence (and prior). Another option is to integrate the hyper-parameters, $\boldsymbol{\theta}$, as shown in Equation 2.20 to obtain the predictive distribution, $p(y^*|\mathbf{x}^*, D)$. The mean of distributions, $p(y^*|\mathbf{x}^*, D, \hat{\boldsymbol{\theta}})$ or $p(y^*|\mathbf{x}^*, D)$ are compared to the true observations at test points, $\mathbf{x}^*$. It is of interest to evaluate and vaildate the quality of the predictions made by GP models. Different validation measures for probabilistic models shall be reviewed.

The Squared Error (SE) is the simplest way of evaluating the quality of a GP model by comparing the model's mean response, $E(y_i^*|\mathbf{x}_i^*, D)$, and true response, y_i^*, using the squared distance loss, $(y_i^* - E(y_i^*|\mathbf{x}_i^*, D))^2$ for test data input, $\mathbf{x}_i^*$. This measure is taken at every test data input, and the average of all SE's form the mean squared error (MSE)

Squared Error

Mean Squared Error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i^* - E(y_i^*|\mathbf{x}_i^*, D))^2,$$

where this assesses the predictive accuracy of the GP based on the mean alone. The measure however is sensitive to the overall scale of the target values, hence this term can be normalized by the variance of the targets over the test data points, to obtain the Standardized Mean Squared Error (SMSE) (Bishop 2006). However, the SMSE or MSE does not take into consideration the predictive variance, $Var(y_i^*|\mathbf{x}_i^*, D)$, and therefore is not taking into account the uncertainty associated with the model. When comparing two models, M_i and M_j, the model with the lowest MSE/SMSE would be the better model when taking into consideration just the predictive mean of the model.

Negative Log Posterior Density

The Negative Log Predictive Density (NLPD) takes into consideration both the predictive mean and variance, $\sigma_{i,*}^2 = Var(y_i^*|\mathbf{x}_i^*, D)$. This measure assumes that predictive density $p(y_i^*|\mathbf{x}_i^*, D)$ is a univariate Gaussian distribution. Calculating this measure across all test points, and averaging leads to the NLPD which is of the form,

$$NLPD = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2n}\log(2\pi\sigma_{i,*}^2) + \frac{(y_i^* - E(y_i^*|\mathbf{x}_i^*, D))^2}{2\sigma_{i,*}^2},$$

where this measure penalizes incorrect variance estimates (Bishop 2006). However this validation measure ignores predictive correlation between test points, only using the diagonals of the predictive co-variance matrix, C, which contains elements, $C_{ij} = Cov((y_i^*|\mathbf{x}_i^*, D), (y_j^*|x_j^*, D))$ (Boukouvalas 2010). Comparing two models, M_i and M_j, the better model would be the one with the lowest NLPD score, based on NLPD measure. The Dawid score is an extension of the validation score, NLPD, where it takes into consideration the predictive co-variance between all test points (Boukouvalas 2010). By introducing notation, $\boldsymbol{\mu}_*$,

Dawid score

$$\boldsymbol{\mu}_* = \begin{pmatrix} E(y_1^*|\mathbf{x}_1^*, D) \\ E(y_2^*|\mathbf{x}_2^*, D) \\ \vdots \\ E(y_n^*|\mathbf{x}_n^*, D) \end{pmatrix}$$

Dawid score is written as,

$$Dawid\ Score = -\log(|C|) - (\mathbf{y}^* - \boldsymbol{\mu}_*)^T C^{-1}(\mathbf{y}^* - \boldsymbol{\mu}_*)$$

where $\mathbf{y}^*$ represents the true responses corresponding to the test points, $\mathbf{x}_i^*$. The difference between Dawid score for two different models can be seen as a numerical approximation to Bayes' factor (Bastos 2010). Comparing two models, M_i and M_j, the best model would be one with larger Dawid score.

2.7.1 Summary

In this chapter, the use of GPs for regression tasks has been reviewed for inputs which are only continuous, where key aspects have included looking at the form of various distributions such as the predictive density, and optimization over the hyperparameters which can be performed using MAP or ML. This chapter, also reviewed various sampling techniques, which could be used to sample from the posterior distribution, $p(\boldsymbol{\theta}|X, \mathbf{y})$, along with some useful diagnostics, such as batch-means and trace plots. The final part of this chapter, looked at various validation metrics, which are used to access the quality of GP models.

The next chapter will involve reviewing the different types of categorical inputs, along with some already established methodologies that can handle a mixture of continuous and categorical inputs.

Chapter 3

Existing Methods Involving Categorical Inputs

This chapter firstly looks at the different types of categorical variables, followed by descriptions of the current GP models which can handle categorical inputs along with the usual continuous ones.

Goals of this chapter

3.1 Different Types of Categorical Inputs

The use of categorical variables causes a discontinuous relationship between the input variable and the output (Brouwer 2002). Examples of data sets which involve a mixture of both categorical and continuous data include 'Boston housing data' , and Abalone Data sets. The Abalone data sets contain a categorical input which determines the sex of an abalone which can take one of three categorical values, male (M), female (F), or infant (I). The inputs in the Abalone data set are used to determine the age of an abalone (Frank & Asuncion 2010). Boston housing data, contains a categorical input, the Charles River dummy variable, which takes values 0 or 1 depending if the tract bounds the river. The inputs in this data set are used to predict the housing values in Boston suburbs (Frank & Asuncion 2010).

different types of categorical inputs

There are different types of categorical variables, such as *ordinal* and *nominal* which we shall look at in closer detail.

Ordinal variables imply there is a natural ordering between the categories. For example, ratings for a movie, have categories, $\{Very\ Bad, Bad, Reasonable, Good, Very\ Good\}$. Other types of ordinal categorical variables include credit classification. Ordinal variables can be encoded into a numeric variable, which then can be used as a continuous input attribute.

ordinal categorical variables

For some categorical variables such as color which has three categories, $\{red, blue, green\}$, it would be unacceptable to use this type of encoding, because there would be no meaningful correspondence between the original categorical values and their associated encoding, and it would impose an natural ordering between the categorical attributes which do not exist (Brouwer 2002). These types of categorical variables are *nominal*. Gender is another type of nominal categorical variable. An example of a real-life data set with no natural ordering between categories is the abalone data set, which has categories, $\{M, F, I\}$, for categorical input, sex.

nominal categorical variables

Nominal categorical variables can be broken up into two further types, binary and multi-class nominal variables. A binary nominal variable, is a variable which could take one of two states, whereas multi-class nominal variables take more than two states. Binary nominal variables can be coded either taking states 0 or 1 (Brouwer 2002). For instance, lie detectors, detect whether someone can either told the truth or told lies. In this case, the truth would be mapped to 0 and a lie would be mapped to 1. However, for multi-class nominal variables, the 1-out-of-C encoding is used to represent nominal categorical variables. Using 1-out-of-C encoding maps a single nominal categorical variable, into C input attributes. Each of the C categories are represented by their own unique row of zeros and ones, where one component takes a value of one, whilst the rest remain at zero. For instance, the roles of people in a university, may be nominal. Suppose there are three states for this categorical variable,
$\{Student, Lecturer, Researcher\}$, then the 1-out-of-3 encoding can be used to represent each of these states such that,

$$Student \rightarrow (0,0,1) \ , \ Lecturer \rightarrow (1,0,0) \ , \ Researcher \rightarrow (0,1,0)$$

The columns of zeros and ones become input attributes, where they replace the original column which contained the original nominal categorical variables. As the number of nominal input attributes used for the model grows, so does the number of input attributes.

multiple qualitative factors for both ordinal and nominal categorical inputs

There could be cases where data sets have more than one qualitative factor, where each factor has n_i possible categories. There are two possible ways to encode the qualitative factors. The first way involves finding all possible combinations of the qualitative factors, and then mapping each of these combinations to a real number (for ordinal case), or using the 1-out-of-C encoding (for multi-class nominal variables). The second approach involves either mapping each category from the j-th qualitative factor to a real number (for the ordinal case), or using a separate 1-out-of-n_j encodings for each separate qualitative factor (in the nominal setting), where n_j is the number of categories for the j-th qualitative factor.

periodic categorical inputs

Nominal and ordinal qualitative factors are not the only two types of categorical data. For example, the days of the week would be example of periodic data, where using the encoding that is meant for ordinal categories is not practical. Instead categories would be encoded using angles (Brouwer 2002).

Next, some of the existing methods which consider continuous and categorical inputs will be reviewed. These include Treed GPs with Limiting Linear Models (Broderick & Gramacy 2010) and the Hypersphere GP model (Zhou et al. 2010).

3.2 Treed Partitioning

Partition trees are used to represent input-output relationships, where they have efficient divide-and-conquer approach to non stationary regression (Chipman et al. 2002). This technique relies on a binary partitioning of input variables, where this forces axis-aligned partitions, and non-stationary modeling. Partitioning is recursive, so each new partition is a sub-partition of the last one. For example, suppose there

is a two-dimensional input space. In this case, a possible partition rule would be to split the region in half by whether the first variable is above or below its midpoint. The second partition however, could only partition in one of the regions previously partitioned on, dividing the space above (or below), meaning in total there are three partitions (not four).

example of a partition

Partition trees are made up of a hierarchy of nodes, where data points are allocated to a node based on a series of splitting rules. Nodes are classified by three names, root, internal and leaf. Root nodes contain all the data points used in the tree, and each node is a root for the preceding sub-tree. If a node has children (i.e. further splits) then the node is considered as internal or otherwise is known as a leaf node. At these leaf nodes, different models can be fitted accordingly, such as constant, linear or GP models (Chipman et al. 2002). Partition trees offer this complexity of fitting a different curve in each region. An illustration of treed partitioning is shown in figure 3.1.

Different types of nodes

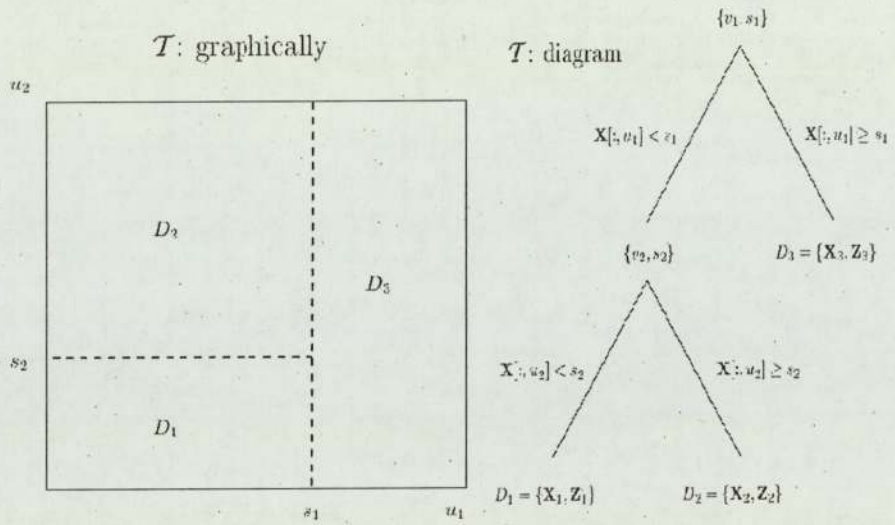


Figure 3.1: An example of a tree T with two splits s_1 and s_2 resulting in 3 partitions as shown in the diagram. D_i represents the data used to build/train models under each leaf.

Tree priors are generative, and specify the tree probability by placing a prior on each individual partition rule. Leaf nodes η may be split with depth-dependent probability of $p_{split}(T, \eta) = \alpha(1 + D_\eta)^\beta$, where β and $\alpha > 0$, are parameters chosen to give an appropriate size and spread to the distribution of trees (Chipman et al. 2002). The co-ordinate $x_{i,j}$ in data point $\mathbf{x}_i$ and location of the split, have an independent prior, which is a uniform distribution over all potential split points. This prior also implicitly, does not allow partitions in the tree space to be created if the resulting region spaces would have very few data points. The posterior distribution over the tree space (Chipman et al. 2002), T, is given by,

Tree probabilities

$$P(T|X, \mathbf{y}) \propto P(\mathbf{y}|X, T)P(T|X). \tag{3.1}$$

Sampling from the tree posterior distribution

In order to sample the posterior distribution (Equation 3.1) of partition trees, and hence explore the posterior space, stochastic changes need to be proposed to the tree structure, T, where these incremental modifications include grow, prune, change and swap (Chipman et al. 2002). These tree evolution moves are accepted based on the Metropolis-Hastings ratio, and have a prior that they are all equally likely moves. Grow and prune moves can add or remove partitions, effectively changing the size of the parameter space, $\boldsymbol{\theta}$, whilst moves like change or swap do not.

Given W leaf nodes $\eta_1, ..., \eta_W$, where leaf node, η_j has parameters $\boldsymbol{\theta}_{\eta,j}$, the leaf likelihood function (after marginalizing over regression model parameters), and assuming the data points are independent is given by,

Leaf likelihood function

$$p(\mathbf{y}|T_t, X) = \prod_{j=1}^{W} \int \prod_{i=1}^{n_j} p(y_{i,\eta_j}|\mathbf{x}_{i,\eta_j}, \boldsymbol{\theta}_{\eta_j}) \pi(\boldsymbol{\theta}_{\eta_j}) \; d\boldsymbol{\theta}_{\eta_j}, \tag{3.2}$$

where y_{i,η_j} represents the observed response at training point, $\mathbf{x}_{i,\eta_j}$ which belong in leaf node η_j under tree T_t. The term n_j represents the amount of training points under leaf node η_j. Alternative notation, includes $[X, \mathbf{y}]^{\eta_j}$ which represents the inputs, and responses belonging to node, η_j (Gramacy & Lee 2008). The predictive distribution for data point, $\mathbf{x}_i^* \in \eta_j$ is given by,

predictive distribution using Treed Models

$$p(y_i^*|\mathbf{x}_i^*, T_t, [X, \mathbf{y}]^{\eta_j}) = \int p(y_i^*|\mathbf{x}_i^*, \boldsymbol{\theta}_{\eta_j}) p(\boldsymbol{\theta}|[X, \mathbf{y}]^{\eta_j}) \; d\boldsymbol{\theta}_{\eta_j}.$$

The complexity of leaves is limited by both computational budget and data dimensions, for example, GP models at the leaves do not allow inference to be integrated out analytically over the model parameters, complicating the posterior inference further (Chipman et al. 2002).

3.2.1 Possible models at leaves

At the leaf nodes on a treed partitioning tree, a choice of different models can be fitted, such as constant or linear mean leaves (Chipman et al. 2002).

Constant Mean Leaves

Given a set of data, $[X, \mathbf{y}]$, where there are a number of M leaf nodes, $\eta_1, \eta_2, ..., \eta_M$. The leaf node, η_i contains c data points, $[X, \mathbf{y}]^{\eta_i}$, and has model parameters, $\mu_{\eta_i}, \sigma^2_{\eta_i}$. The constant mean model assumes that the outputs for data points in node η_i are distributed as,

Constant Mean Leaves

$$y_1, y_2, ..., y_c \sim N(\mu_{\eta_i}, \sigma^2_{\eta_i}).$$

Linear Mean Leaves

Extending constant mean leaf models, another choice of model to fit at the leaves is the linear mean model. The leaf node, η_i contains c data points, $[X, \mathbf{y}]^{\eta_i}$, and has model parameters, $\boldsymbol{\mu}_{\eta_i}, \boldsymbol{\gamma}_{\eta_i}$, where they are $B \times 1$-dimensional. All training inputs $\mathbf{x}_i$ can be collected together to form matrix, X, which is $N \times B$-dimensional. The model, assumes that the responses in node η_i are distributed as,

Linear Mean Leaves

$$y_1, y_2, ..., y_c \sim N(\boldsymbol{\mu}_{\eta_i} + X\boldsymbol{\gamma}_{\eta_i}, \sigma^2_{\eta_i}).$$

3.2.2 Treed Gaussian Processes

Another type of model that can be fitted at the leaves of a partition tree is a Gaussian process. GPs are flexible priors over functions, whereas the trees divide the predictor space based on splitting rules, however when these combined together, forming treed Gaussian processes (TGP) they form a powerful approach to non-stationary regression problems. GPs with linear trends are fit independently to each region, in a hierarchical manner, where each region has its own set of hyper-parameters, which makes the overall process, non-stationary. This particular model is developed by Broderick & Gramacy (2010). Trees are averaged out by integrating over possible trees, using reversible-jump Markov chain Monte Carlo (RJ-MCMC) (Chipman et al. 2002), where the tree prior is specified through a tree-generating process.

Treed Gaussian processes

Treed GPs have been applied in a number of applications in particular, sequential design and analysis of computer experiments (Chipman et al. 2002), and the methodology has been adapted to also include categorical inputs.

application of treed GPs

3.2.3 Categorical Inputs using TGP

Broderick & Gramacy (2010) have recently added to the existing TGP package (which is available as code in R) by allowing categorical inputs to be included as part of the design for inference. Broderick & Gramacy (2010) tried several approaches such as CART, Treed GP with Limiting Linear models and Treed GPs. When categorical inputs are required, they encode them in a binary form. For example, if N categories, $S_1, ..., S_N$ were used in the design, they would have a boolean representation,

Encoding

$$S_i = (0, ..., 1\delta_{S_i}, ..., 0) \in \mathbb{R}^{N-1},$$

where the final category, $S_N = (0, ..., 0) \in \mathbb{R}^{N-1}$. Broderick & Gramacy (2010) started the investigation by using a Treed GP with a limiting linear model, ignoring the categorical inputs, however this gave rise to a high root mean squared error (RMSE), hence the categories needed to be included. The use of Bayesian CART with the inclusion of the categorical inputs, did not partition on the categorical inputs correctly. The suggestion of using a Treed GP with categorical inputs would lead to rank-deficiency of the design input matrix, where there would a column of zeros or ones. Broderick & Gramacy (2010), then added the functionality of only allowing the continuous inputs to predict the responses under the GPs at the leaves, whilst both the categorical and continuous inputs are still candidates for treed partitioning. However, Broderick & Gramacy (2010) then argue that using only the categorical inputs for partitioning, would have the benefit of improved mixing in the Markov chain. The model used in this case is the Treed GP with limiting linear models. However, if the number of categories increase, this methodology can become increasingly slow.

Different Approaches tried before coming to a solution

Broderick & Gramacy (2010) also questions whether it is worth-while and possible to design a GP correlation function which can explicitly handle both a mixture of qualitative (categorical) and quantitative (real-valued).

possibility of having GP correlation handling categorical inputs

3.3 Hypersphere GP Model

Zhou et al. (2010) proposes a flexible approach for building Gaussian process models where the data inputs have both quantitative and qualitative factors, using a Hypersphere parameterization to model the correlations of the qualitative factors, where the qualitative factors are assumed to be nominal. The benefit realized by

restrictive and non-restrictive correlation functions

Zhou et al. (2010) avoids the need to directly solve optimization problems with positive definite constraints, and uses a unrestrictive 'structure-free' correlation function between the qualitative factors. This work builds on previous work of Qian et al. (2008) which considered optimization techniques in semi-definite programming for ensuring positive constraints on the correlation matrix, T, when maximizing the likelihood with this particular constraint. It is possible to simplify the complexity of the work shown in the paper of Qian et al. (2008) by using restrictive correlation functions for the qualitative factors: however this cannot capture the various types of correlations between the categories. The parameterization suggested by Zhou et al. (2010) allows for both positive and negative correlations between qualitative factors to be captured in the GP model. The two different models shown in the papers Zhou et al. (2010) and Qian et al. (2008) are inter-connected, where they fit kriging type models with both qualitative and quantitative factors however having different degrees of flexibility.

Suppose there is a set of training data $D = \{(\mathbf{x}_i, y_i), i = 1, ..., n\}$, where $\mathbf{x}_i$ is a mixture of n_c qualitative and n_d quantitative factors as shown below,

$$\mathbf{x}_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,n_d} \\ w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,n_c} \end{pmatrix}$$

The vectors $\mathbf{x}_{i,c}$ and $\mathbf{x}_{i,d}$ contain only the quantitative or qualitative factors for data input, $\mathbf{x}_i$.

$$\mathbf{x}_{i,c} = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,n_d} \end{pmatrix} \quad \mathbf{x}_{i,d} = \begin{pmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,n_c} \end{pmatrix}$$

qualitative and quantitative factors

The co-variates, $x_{i,j}, j = 1, ..., n_d$, are the quantitative factors, and $w_{i,j}, j = 1, ..., n_c$, are the qualitative factors for data point, $\mathbf{x}_i$. Qualitative factors, $w_{i,j}$ can take a state from one component of the set, A_j,

$$A_j = \{A_{j,1}, A_{j,2}, ..., A_{j,n_{c,j}}\},$$

where $A_{j,k}$ represents the k-th category for the j-th qualitative factor, and $n_{c,j}$ represents the number of categories for the j-th qualitative factor. Test points, $\mathbf{x}_i^*$ are assumed to have the same form as the training inputs. The response at input, $\mathbf{x}_i$ is modeled as,

$$y(\mathbf{x}_i) = m(\mathbf{x}_i)\boldsymbol{\beta} + Q(\mathbf{x}_i),$$

where $m(\mathbf{x}_i)$ is the mean function evaluated at input, $\mathbf{x}_i$ which is multiplied by the vector of unknown coefficients, $\boldsymbol{\beta}$. $Q(\mathbf{x}_i)$ is a Gaussian process with mean 0 and process variance σ_p^2.

different forms of kernel

There are many possible forms for the kernel, k, between the quantitative factors, that include Matern and exponential kernels, and whether these kernels are isotropic

or separable (Rasmussen 2006, p.79). However Zhou et al. (2010), uses the (popular) squared exponential correlation function which is given by,

$$k(\mathbf{x}_{a,c}, \mathbf{x}_{b,c}) = \exp\left(\sum_{j=1}^{n_d} \frac{1}{\phi_i}(x_{a,j} - x_{b,j})^2\right),$$

where ϕ_i is the length scale for the i-th continuous covariate. Suppose that $\mathbf{x}_{i,d}$ is 1-dimensional (hence there is one qualitative factor). The co-variance evaluated at each pair of inputs, $\mathbf{x}_a$, and $\mathbf{x}_b$ will be of the form,

$$cov(\mathbf{x}_a, \mathbf{x}_b) = \sigma_p^2 T(w_{a,1}, w_{b,1}) K(\mathbf{x}_{a,c}, \mathbf{x}_{b,c}),$$

positive definite T

where $T(w_{a,1}, w_{b,1}) = T(w_{b,1}, w_{a,1})$ are the cross-correlation parameters between categories $w_{a,1}$ and $w_{b,1}$. The values of $T(A_{1,i}, A_{1,j})$, $\forall i, j$, are stored in a positive definite matrix, T, with unit diagonal elements (known as PDUDE). Both, Qian et al. (2008) and Zhou et al. (2010) use this approach to modeling the categorical inputs, but differences between their methodologies, is the way correlation parameters, $T(.,.)$ areconstructed. Zhou et al. (2010) decompose matrix, T, using a Cholesky-type decomposition, where $T = LL^T$. L is a lower triangular matrix, where the diagonal entries are strictly positive. Each row in L, i,e. $(l_{r,1}, l_{r,2}, ..., l_{r,r})$ is modeled as co-ordinates of a surface point on an r-dimensional unit hypersphere. In order to satisfy this constraint, the spherical co-ordinate system must be used as follows,

decomposition of T

$$l_{1,1} = 1,$$

for $s = 2, ..., r-1$,

$$l_{r,s} = \sin(\theta_{r,1}) ... \sin(\theta_{r,s-1}) \cos(\theta_{r,s}),$$

$$l_{r,r} = \sin(\theta_{r,1}) ... \sin(\theta_{r,r-2}) \cos(\theta_{r,r-1}),$$

and for $p = 2, ..., r$,

$$l_{p,1} = \cos(\theta_{r,1})$$

where $\theta_{r,s} \in (0, \pi)$, $\forall s < r$, known as box constraints (Zhou et al. 2010). This parameterization ensures that T is positive-definite because restriction of each $\theta_{r,s}$, ensures the $l_{r,r}$ (diagonal elements of matrix L) are strictly positive, and this ensures that T is a positive-definite matrix. Elements of the PDUDE matrix T can be obtained component wise for each $t_{i,j}$ where $i \leq j$ using the following equations,

constraints for matrix, T

$$t_{i,j} = \sum_{p=1}^{min(i,j)} l_{i,p} l_{p,j}, \quad \forall i, j > 1 \tag{3.3}$$

$$t_{j,1} = l_{j,1}, \quad \forall j > 2 \tag{3.4}$$

$$t_{j,j} = 1. \tag{3.5}$$

Because T is also symmetric, $t_{i,j} = t_{j,i}$, $\forall i, j$. This parameterization also ensures that $t_{i,i} = 1$, $\forall i$. The use of this parameterization allows the elements in T to either be positive or negative, and hence can capture positive and negative correlations between categories. The number of parameters for constructing PDUDE matrix T is $\frac{n_c(n_c-1)}{2}$.

capture correlations between categories

multiple qualitative factors

The variable, $n_{c,j}$ indicates the number of possible states of the j-th qualitative input. There are two choices here. Firstly, one could collect all data points, $\mathbf{x}_i$, which have the same qualitative factors, and place these data points in the same category. The number of different possible combinations here would indicate the total number

of categories to be used in the algorithm (Zhou et al. 2010).

grouping qualitative factors

In this case, one could use the methodology used when first discussing the Hypersphere parameterization model with one qualitative factor with B number of possible categories. The second choice, is to use a separate PDUDE matrix, T, for each qualitative factor using the following kernel between data points, $\mathbf{x}_a$ and $\mathbf{x}_b$,

separate T matrix per qualitative factor

$$cov(\mathbf{x}_a, \mathbf{x}_b) = \prod_{i=1}^{n_c} \sigma_p^2 T(w_{a,i}, w_{b,i}) k(\mathbf{x}_{a,c}, \mathbf{x}_{b,c}).$$

differences between both approaches

One of the major differences between using these two different approaches for the case where multiple qualitative factors are part of the input design, is the number of parameters required for constructing the PDUDE matrix(es), T. Suppose by combining data with the same qualitative factors, there are R possible combinations, then $\frac{R(R-1)}{2}$ parameters are needed to construct the single PDUDE matrix, T, whereas the other approach, which uses n_c separate PDUDE matrices, requires $\sum_{i=1}^{n_c} \frac{(n_{c,i})(n_{c,i}-1)}{2}$ parameters.

optimization of the hyper-parameters

ML or MAP can be performed to estimate values for the unknown parameters, $\boldsymbol{\beta}$, $\theta_{r,s}, r < s$, σ_p^2, and $\phi_i, i = 1, ..., n_d$. This problem can be solved using a standard non-linear optimization algorithm. However, Zhou et al. (2010) uses various models and implementations offered by Qian et al. (2008), as a way of finding some good initial starting parameters for the inverse length scales $(\phi_i, i = 1, ..., n_d)$, to be used in the Hypersphere GP model.

Other Existing ways of dealing with categorical inputs discussed briefly

These are the two main methodologies which have been reviewed, however they are not the only ways in the literature of using categorical inputs in a model. Hannah et al. (2011) combines Dirichlet processes (DP) with generalized linear models (DP-GLMs), where categorical covariates are modeled by a mixture of multinomial distributions, and the count response by a Poisson distribution. An advantage with using DP-GLMs is that it is able to capture hetroscedasticity, where the noise variance is input-dependent, whereas GPs make assumptions about the data dispersion and homoscedasticity (Hannah et al. 2011). Another possible way of dealing with categorical inputs is to build a separate independent GP for each combination of the qualitative factors (Zhou et al. 2010).

In the next chapter, we will demonstrate our proposed methodology which can handle data with categorical inputs.

Chapter 4

Embedded Gaussian Process

Goal of this chapter

This chapter, discusses in detail our proposed GP which can handle data with categorical inputs. This GP will be referred to as the Embedded GP. Two types of categorical variables, *nominal* and *ordinal*, will be considered in this framework. This chapter will later compare the Embedded GP with existing methodologies in the literature, which include the Treed GP and Hypersphere GP models that were reviewed in the previous chapter. The final component of this chapter, will give the user some useful guidelines to follow when using the Embedded GP in practice, along with various implementations of the Embedded GP which were used in the next chapter.

4.1 Different Types of Categorical Variables

two types of categorical variables

GPs are priors over the function values, $\mathbf{f} \sim GP(m(\mathbf{x}_i), k(\mathbf{x}_i, \mathbf{x}_j))$, with mean function, $m(.)$ and co-variance matrix, K, where data inputs, $\mathbf{x}_i$ contain both continuous and categorical elements (Rasmussen 2006, p.13). Data points, $\mathbf{x}_i$ contain a mixture of n_c qualitative and n_d quantitative factors as shown below,

$$\mathbf{x}_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,n_d} \\ w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,n_c} \end{pmatrix}$$

The vectors, $\mathbf{x}_{i,c}$ and $\mathbf{x}_{i,d}$, contain only the quantitative or qualitative factors for data input, $\mathbf{x}_i$.

$$\mathbf{x}_{i,c} = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,n_d} \end{pmatrix} \quad \mathbf{x}_{i,d} = \begin{pmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,n_c} \end{pmatrix}$$

The co-variates $x_{i,j}, j = 1, ..., n_d$, are the quantitative factors, and $w_{i,j}, j = 1, ..., n_c$, are the qualitative factors for data point, $\mathbf{x}_i$. Qualitative factors, $w_{i,j}$ can take a state from one component of the set, A_j,

$$A_j = \{A_{j,1}, A_{j,2}, ..., A_{j,n_{c,j}},\}$$

where $A_{j,k}$ represents the k-th category for the j-th qualitative factor, and $n_{c,j}$ represents the number of categories for the j-th qualitative factor. However categorical elements cannot be directly placed into the GP as they are.

We shall proceed to explain the methodology behind the Embedded GP when there is one qualitative factor with C categories, for the two different scenarios.

4.2 Embedded GP for Ordinal Categorical Inputs

assigning categories to numbers on 1-D axis

Let us start off with explaining the case of one qualitative factor with C possible ordinal categories. Categories which are ordinal assume a natural ordering, where these categories are assigned to a number on a one-dimensional axis. We propose mapping each of the categorical variables $A_{1,j}$, $\forall j \leq n_{c,1}$,

$$A_1 = \{A_{1,1}, A_{1,2}, ..., A_{1,n_{c,1}}\} \rightarrow g_1 = \{g_{1,1}, g_{1,2}, ..., g_{1,n_{c,1}}\}$$

to a number, $g_{1,j}$ respectively on a real number line. An example of this is shown in Figure 4.1. Data points, $\mathbf{x}_i$ will be transformed accordingly,

$$\mathbf{x}_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,n_d} \\ w_{i,1} = A_{1,k} \end{pmatrix} \rightarrow \mathbf{x}_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,n_d} \\ g_{1,k} \end{pmatrix}$$

other parameters in the model

where $w_{i,1}$ is the qualitative factor which takes the k-th value from the set of $n_{c,1}$ possible categories. However, the Embedded GP will also have other parameters, which are contained in the mean and kernel functions. Such parameters include the individual length scales (for separable kernels) or a single length scale (for isotropic kernels), noise and a single signal variance.

how categorical encodings could be learnt

All parameters, including the categorical encodings, $g_{1,k}$, $\forall k \leq n_{c,1}$ are learnt by maximizing the log posterior distribution, $\log(p(\boldsymbol{\theta}|X, \mathbf{y}))$, where now $\boldsymbol{\theta}$ additionally contains the categorical encodings. Once optimal values for $\boldsymbol{\theta}$ have been obtained, the optimal categorical mappings, $g_{1,k}$, $\forall k \leq n_{c,1}$ are used for test data points, $\mathbf{x}_i^*$, such that,

$$\mathbf{x}_i^* = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ ... \\ x_{i,n_d} \\ w_{j,1} = A_{1,k} \end{pmatrix} \rightarrow \mathbf{x}_i^* = \begin{pmatrix} x_{i,1} \\ w_{i,2} \\ ... \\ x_{i,n_d} \\ \hat{g_{1,k}} \end{pmatrix}$$

existing kernels used in GP emulation

The rest of the hyper-parameters are used in the predictive equation, $p(y_i^*|\mathbf{x}_i^*, X, \mathbf{y})$. Some commonly used isotropic and stationary kernel functions include the Matern and squared exponential co-variance functions. These kernels, depend on term r which measures the Euclidean distance between data points, $\mathbf{x}_i$, $\mathbf{x}_j$. For ordinal categorical variables, r^2 is given by

euclidean distance between points

$$r^2 = \sum_{p=1}^{n_d} \frac{1}{\phi_p}(x_{i,p} - x_{j,p})^2 + (g_{1,k_1} - g_{1,k_2})^2,$$

for the case of per continuous input length scales, where data points, $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to categories, A_{1,k_1} and A_{1,k_2} respectively and $\phi_p, p = 1, ..., n_d$, are the inverse

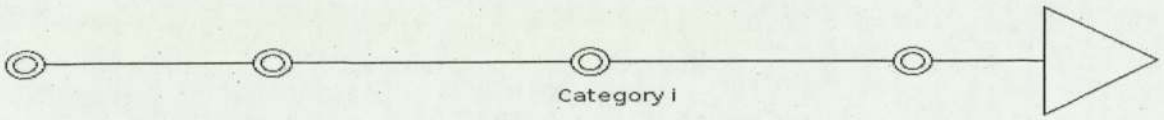


Figure 4.1: Example of placing the categorical mappings, $g_{1,j}$ onto the real number axis (ordinal case).

length scales for the p-th continuous covariate. Length scales can also be applied to the categorical co-variate, however this would introduce coupling between the length scales (ϕ) and the mapped categorical mappings.

relevance of inputs

The advantage of having a separate length scale per continuous co-variate is that they determine how relevant a continuous input is, and if the length scale is very large, then the co-variance will become almost independent of that input (Bishop 2006, p.106-107).

For the case where there are multiple qualitative factors such that data inputs, $\mathbf{x}_i$ have the form,

$$\mathbf{x}_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,n_d} \\ w_{i,1} = A_{1,k_1} \\ w_{i,2} = A_{2,k_2} \\ \vdots \\ w_{i,n_c} = A_{n_c,k_{n_c}} \end{pmatrix}$$

multiple qualitative factors

where the s-th qualitative factor, $w_{i,s}$, takes a state from the set A_s. Suppose there are q combinations of qualitative factors; each combination would be assigned to a real number $g_{1,m}$, where $m \in [1, q]$ and then inference would be done, as for one qualitative factor.

4.3 Embedded GP for Nominal Categorical Inputs

For nominal categorical variables, the data inputs entered in the Embedded GP model would be different because these variables do not have a natural ordering. We shall discuss the Embedded GP model where there is one nominal qualitative variable before discussing inputs which contain multiple categorical nominal variables.

1-out-of-C encoding

In this case, one qualitative factor may have C possible categories, but these do not impose a natural ordering. Instead each of the categories, $A_{1,k}$ $\forall k \leq n_{c,1}$ are encoded using the 1-out-of-C encoding where each category would be assigned its own unique zeros and ones. Hence for $k = 1, ..., n_{c,1}$,

$$A_1 = \{A_{1,1}, A_{1,2}, ..., A_{1,n_{c,1}}\},\ A_{1,k} \rightarrow Y_k = (g_{1,1}\delta_1, ..., g_{1,k}\delta_k, ..., g_{1,n_{c,1}}\delta_{n_{c,1}}) \in \mathbb{R}^{n_{c,1}}$$

binary representations multiplied

where $\delta_k = 1$ represents the fact, that data point, $\mathbf{x}_i$ belongs to category k, whilist for other terms, $\delta_j = 0$, $j = 1, ..., n_{c,1}$, $j \neq k$. The role of the Embedded GP is to determine the distance of each category, $A_{1,i}$, $i = 1, ..., n_{c,1}$ from the origin, where an example is shown in Figure 4.2. Data points, $\mathbf{x}_i$,

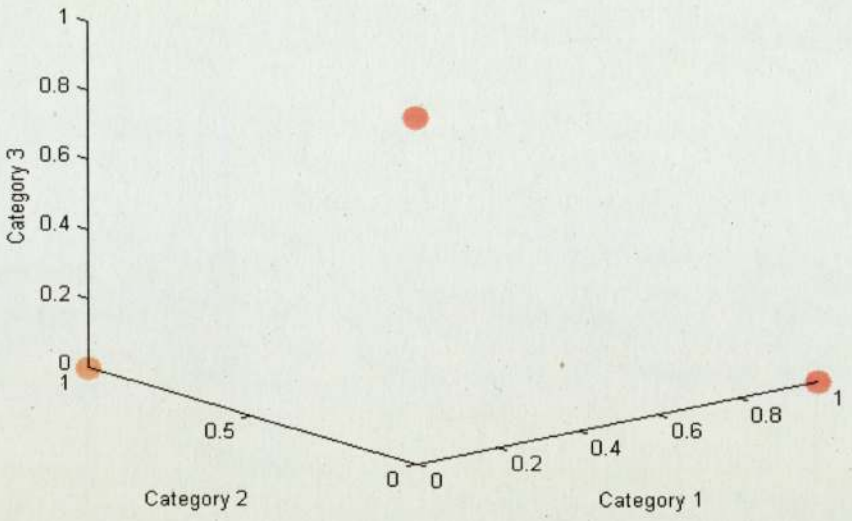


Figure 4.2: Demonstration of the case where one qualitative factor with 3 categorical values are used. Each categorical value is mapped into a $\mathbb{R}^3$ space, where each category is assigned an axis. The distances between the category mappings and the origin in this example is one.

$$\mathbf{x}_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,n_d} \\ w_{i,1} = A_{1,k} \end{pmatrix}$$

are converted to the form,

$$\mathbf{x}_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,n_d} \\ g_{1,1}\delta_1 \\ g_{1,2}\delta_2 \\ \vdots \\ g_{1,k}\delta_k \\ \vdots \\ g_{1,n_{c,1}}\delta_{n_{c,1}} \end{pmatrix}$$

optimizing the hyper-parameters using ML/MAP

which are used in the Embedded GP algorithm (for nominal categorical variables). However, just like the ordinal case, the optimum values for each of the distances, $g_{1,j}$, $j = 1, ..., n_{c,1}$, is not clear, and is determined along with the length scales, noise and signal variances to maximize the log posterior distribution. After obtaining these optimal values for the hyper-parameters, these are placed in the predictive distribution. For (some) isotropic stationary kernels, as mentioned above, the Euclidean distance between two arbitrary data points, $\mathbf{x}_i$ and $\mathbf{x}_j$ are needed. The squared distance, r^2, between these points have two scenarios, whether both data points belong to the same category or whether they belong to the different categories. For data points belonging to the same category, $A_{1,j}$, the squared distance, r^2, between these

distances between data points

points is,

$$r^2 = \sum_{p=1}^{n_d} \frac{1}{\phi}(x_{i,p} - x_{j,p})^2, \tag{4.1}$$

and the squared distance, r^2, between two data points, belonging to different categories A_{1,k_1} and A_{1,k_2}, where $1 \leq k_1, k_2 \leq n_{c,1}$ is,

$$r^2 = \sum_{p=1}^{n_d} \frac{1}{\phi_p}(x_{i,p} - x_{j,p})^2 + (g_{1,k_1}^2) + (g_{1,k_2}^2), \tag{4.2}$$

where both Equations 4.1 and 4.2 assume a separate length scale for each continuous input. Data points may have many nominal qualitative factors in more complex data sets. One approach of handling data points in this manner for the Embedded GP in the nominal setting, is to assign each combination of qualitative factors to their own unique row of zeros and ones, and proceed similarly to the case of having one nominal qualitative factor.

multiple qualitative factors in data points

4.4 Hyper-parameters used in the Embedded GP

The Embedded GP for either types of categorical variables, *nominal* or *ordinal*, uses hyper-parameters, $\boldsymbol{\theta}$, which contain, σ_n^2 (noise variance), σ_p^2 (process variance), $\phi_i, i = 1, ..., n_d$ (separable kernel) or ϕ (isotropic kernel) and the categorical one-to-one mappings, $g_{1,j}, j = 1, ..., n_c$, where n_d and n_c represent the number of continuous and categorical inputs respectively (for one qualitative factor). The Embedded GP model assumes that the same levels of qualitative factors (where there are n_c levels) are treated as categories in a one qualitative factor setting discussed above. Other additional hyper-parameters which appear in the GP prior will be explicitly stated in later chapters.

hyper-parameters which the Embedded GP uses

4.5 Relationship between Embedded GP and Hypersphere GP Models

We have seen several ways of handling categorical inputs using GP models, such as the Embedded GP and Hypersphere GP models. These models seem similar. The difference between the two models, is that our embedded GP approach considers placing the categorical one-to-one mapping values, $g_{1,j}, j = 1, ..., n_c$ as part of the data input, which is used in the distance function r in the co-variance between the two data points. The Hypersphere GP model multiplies the kernel, k by a correlation term that depends on the categories of the two inputs, $\mathbf{x}, \mathbf{x}'$. It is possible that these two models could be equivalent. Consider, the Embedded GP for the nominal case, where this GP and the Hypersphere GP models have one qualitative factor, with n_c possible categories. We shall use the Gaussian correlation kernel which is of the following form,

differences between the models

$$k(\mathbf{x}_1, \mathbf{x}_2) = e^{-\sum_{i=1}^{n_d} \frac{1}{\phi_i}(x_{1,i} - x_{2,i})^2},$$

for continuous inputs, to investigate the possible equivalence between the two models. Suppose that $\mathbf{x}_1$ and $\mathbf{x}_2$ belong to categories A_{1,k_1} and A_{1,k_2} respectively. All n_c categories $A_{1,j}, j = 1, ..., n_c$, are mapped onto an n_c-dimensional space, with co-ordinate, $(0, ..., \delta_j, ..., 0)$ which is then multiplied by term, $g_{1,j}$, which scales the j-th dimension whilst the other dimensions remain zero. The Embedded GP (for the

nominal case) uses the following kernel, k, if $\mathbf{x}_1$ and $\mathbf{x}_2$ do not belong to the same categories,

$$k(\mathbf{x}_1, \mathbf{x}_2) = \sigma_p^2 e^{-(\sum_{i=1}^{n_d} \frac{1}{\phi_i}(x_{1,i}-x_{2,i})^2 + g_{1,k_1}^2 + g_{1,k_2}^2)} + \sigma_n^2$$

and if the data points belong to the same category then,

$$k(\mathbf{x}_1, \mathbf{x}_2) = \sigma_p^2 e^{-\sum_{i=1}^{n_d} \frac{1}{\phi_i}(x_{1,i}-x_{2,i})^2} + \sigma_n^2,$$

where σ_p^2, and σ_n^2 represent the signal and noise variances respectively, and n_d represents the amount of continuous factors. The Hypersphere GP model, will instead use the following kernel,

$$k(\mathbf{x}_1, \mathbf{x}_2) = \sigma_p^2\, T(A_{1,k_1}, A_{1,k_2})\, e^{-\sum_{i=1}^{n_d} \frac{1}{\phi_i}(x_{1,i}-x_{2,i})^2} + \sigma_n^2,$$

where $T(A_{1,k_1}, A_{1,k_2})$ is a term which depends on categories A_{1,k_1} and A_{1,k_2}. Before proceeding, consider the term $g_{1,k_1}^2 + g_{1,k_2}^2$ equating to a single-term $p_{A_{1,k_1},A_{1,k_2}}$, which depends on categories A_{1,k_1} and A_{1,k_2}.

possible equivalence

One possible way the models would be equivalent, would be to force the hyperparameters, σ_n^2, σ_p^2, $\phi_i, i = 1, ..., n_d$ which appear in the kernels of both models to be the same. Consider the case where two data points $\mathbf{x}_1$ and $\mathbf{x}_2$ belong to the same category, A_{1,k_1}. Hence we assert,

$$\sigma_p^2 e^{-\sum_{i=1}^{n_d} \frac{1}{\phi_i}(x_{1,i}-x_{2,i})^2} + \sigma_n^2 \;?=\; \sigma_p^2\, T(A_{1,k_1}, A_{1,k_2})\, e^{-\sum_{i=1}^{n_d} \frac{1}{\phi_i}(x_{1,i}-x_{2,i})^2} + \sigma_n^2.$$

The Hypersphere GP model requires that the correlation term $T(.,.)$ between the same categories is one. Therefore,

$$\sigma_p^2 e^{-\sum_{i=1}^{n_d} \frac{1}{\phi_i}(x_{1i}-x_{2i})^2} + \sigma_n^2 \;=\; \sigma_p^2\; e^{-\sum_{i=1}^{n_d} \frac{1}{\phi_i}(x_{1i}-x_{2i})^2} + \sigma_n^2.$$

Trivially, for this case the two approaches are equivalent. Now let us investigate the case where data points $\mathbf{x}_1$ and $\mathbf{x}_2$ belong to two distinct categories. Hence,

$$\sigma_p^2 e^{-(\sum_{i=1}^{n_d} \frac{1}{\phi_i}(x_{1,i}-x_{2,i})^2 + (g_{1,k_1}^2 + g_{1,k_2}^2))} + \sigma_n^2 \;?=\; \sigma_p^2\, T(A_{1,k_1}, A_{1,k_2})\, e^{-\sum_{i=1}^{n_d} \frac{1}{\phi_i}(x_{1,i}-x_{2,i})^2} + \sigma_n^2.$$

Through cancellation,

$$e^{-(\sum_{i=1}^{n_d} \frac{1}{\phi_i}(x_{1,i}-x_{2,i})^2 + g_{1,k_1}^2 + g_{1,k_2}^2)} \;?=\; T(A_{1,k_1}, A_{1,k_2})\, e^{-\sum_{i=1}^{n_d} \frac{1}{\phi_i}(x_{1,i}-x_{2,i})^2}.$$

Then using the rule of exponentials, $e^{-x-y} = e^{-x}e^{-y}$, hence

$$e^{-\sum_{i=1}^{n_d} \frac{1}{\phi_i}(x_{1,i}-x_{2,i})^2} e^{-(g_{1,k_1}^2 + g_{1,k_2}^2)} \;?=\; T(A_{1,k_1}, A_{1,k_2})\, e^{-\sum_{i=1}^{d_c} \frac{1}{\phi_i}(x_{1i}-x_{2i})^2}.$$

And therefore,

$$e^{-(g_{1,k_1}^2 + g_{1,k_2}^2)} \;?=\; T(A_{1,k_1}, A_{1,k_2}).$$

By taking logs of both sides, yields

$$-(g_{1,k_1}^2 + g_{1,k_2}^2) \;?=\; \log(T(A_{1,k_1}, A_{1,k_2})).$$

possible equivalence

Because, g_{1,k_1}^2 and g_{1,k_2}^2 are both positive functions, therefore $T(A_{1,k_1}, A_{1,k_2})$ needs to be less than or equal to 1. The Hypersphere GP model requires that $T(.,.)$ is in the interval $(-1, 1)$, whereas the Embedded GP accepts, $e^{-(g_{1,k_1}^2 + g_{1,k_2}^2)}$ to be in the range $(0, 1)$. Therefore, it is possible for the two approaches to be equivalent if there

is a solvable linear system of equations, with the limitation that positive correlations are permitted between all categories.

It may be possible to prove that the two approaches, Hypersphere GP model and the Embedded GP model are equivalent with other kernels, but we leave this to further work.

possible kernels could be equivalent

4.6 Differences Between the Embedded GP and Other Existing Methodologies For Categorical Inputs

We have reviewed two methodologies which can handle GPs with both categorical and continuous inputs: Treed GPs with the Linear Limiting Model; and the Hypersphere GP model. In addition, we have added to these methodologies, by designing a GP which can also handle data points with continuous and categorical elements, by either mapping each category to a real number (for the ordinal case), or using the 1-out-of-C encoding (for the nominal case).

current existing GP models handling categorical inputs

Treed GPs with linear limiting models, do not explicitly place the categorical variables into the covariance function, $k(\mathbf{x}_i, \mathbf{x}_j)$, for the GPs at the leaves of the tree, but rather the binary representation of these categories are used for treed partitioning, and the continuous inputs are used at the leaves. This method has the disadvantage of not being able to fit GPs containing categorical inputs, and does not consider the correlation between these type of inputs. This method also relies on sampling the posterior distribution, $P(T|X, \mathbf{y})$ using RJ-MCMC, where this will not be able to search through all possible optimal trees. Broderick & Gramacy (2010) questions whether it possible to design a kernel which contains both these categorical and continuous inputs. The Hypersphere GP model is able to perform this task.

Treed GP comparisons

The Hypersphere GP model has the advantage of being to model negative and positive correlations between categories using an unrestrictive correlation between them, thus avoiding the need of directly solving optimization problems with positive definite constraints.

Hypersphere GP comparisons

The proposed GP model, Embedded GP however is able to handle cases of nominal or ordinal categorical variables for each qualitative factor. However, when both of these models, use the squared exponential kernel, the Embedded GP is more restricted than the Hypersphere GP model since the Embedded GP is not able to handle negative correlations.

Embedded GP comparisons

The number of parameters required for these two GP models differ; the Embedded GP model requires d parameters for each of the corresponding mappings from categories to reals, whereas the Hypersphere GP model requires at least $\frac{d(d-1)}{2}$ for the correlation parameters between each pair of categories. As the number of categories grow, the more parameters the Hypersphere GP model will need to estimate compared to the Embedded GP.

parameter difference between Hypersphere GP and Embedded GP models

The number of independent GPs fitted through data for each combination of qual-

independent GP model comparisons

itative factors (or category), will also grow as the number of categories increase. Building independent GPs per category, ignores the correlations between the responses from each category. These correlations can vary between values of -1 and 1. Negative correlation indicates that a given response of one category is increasing whilist the response from a different category is decreasing. Positive correlation occurs when either the responses from both categories are increasing, or both of them have responses which are decreasing. In some situations, considering these correlations could be important.

Number of GP models

The Hypersphere GP model and Embedded GP will use a single GP whereas using the Treed GPs or Independent GPs per category, will require multiple GPs.

4.7 Implementation and Efficiency

This section outlines a general implementation of the Embedded GP model and contains suggestions on how to improve the computing performance of this GP.

4.7.1 Efficiency

Below are some suggestions in order to improve the efficiency of the Embedded GP model (in both the nominal and ordinal setting),

1. Rescale the data inputs to lie in the interval, $[0, 1]$, where this will ensure better conditioning. This is global, so do not rescale inputs per category.
2. Make sure the continuous data inputs, $\mathbf{x}_i$ (preferably for both training and testing) are different for each category (or combinations of qualitative factors), otherwise, this can lead to principle minors in the matrix, K having a determinant of 0. An example is shown in Appendix A.
3. Use Cholesky decomposition whenever the inverse of matrices are required, for example, GP-models require the inversion of the co-variance matrix, K.
4. Some hyper-parameters, such as the length scales, signal and noise variances are required to be positive (hence ≥ 0). The posterior or likelihood distributions need to be constrained taking this into consideration, where the parameters are constrained to take the form, $\exp^{-m}$.

4.7.2 General Implementation of the Embedded GP Model

In this section, we outline the general steps to be performed when using the Embedded-GP model in practice. For more detailed implementations for specific cases, please read Appendix B. The implementations discussed in Appendix B have been used for experimentation in Chapter 5.

We consider training and test data, $(X, \mathbf{y})$ and $(X^*, \mathbf{y}^*)$ which has n_d continuous co-variates and n_c categories (or n_c different levels of qualitative factors). Also assuming a GP prior of $GP(m, k)$, following steps are performed when using the Embedded GP for data inputs, $\mathbf{x}_i$ which contain j_D nominal qualitative factors,

1. Given n_c different combinations of qualitative factors, assign each combination to a hyper-parameter $g_{1,j}$.

This transforms data input, $\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n_d}, w_{i,j_1}, w_{i,j_2}, ..., w_{i,j_D})' \rightarrow \mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n_d}, v_1, v_2, ..., v_{n_c})'$.
Vector $\mathbf{v} = (v_1, v_2, ..., v_{n_c}) = (g_{1,1}\delta_1, g_{1,2}\delta_2, ..., g_{1,j}\delta_j, ..., g_{1,n_c}\delta_{n_c})$.

The transformed input will be used in the Embedded GP. The notation, $\delta_j = 1$ if the data point belongs to category j whilst the other variables, $\delta_k = 0, k = 1, ..., n_c, k \neq j$.

Example. Suppose there are two qualitative factors, A_1 and A_2. Each qualitative factor has $n_{c,1} = 2$ and $n_{c,2} = 2$ categories respectively. Hence,

$$A_1 = \{A_{1,1}, A_{1,2}, ..., A_{1,n_{c,1}}\},$$

$$A_2 = \{A_{2,1}, A_{2,2}, ..., A_{2,n_{c,2}}\}.$$

The possible combinations with these categories are,

$$C_1 = \{A_{1,1}, A_{1,2}\}, \quad C_2 = \{A_{1,1}, A_{2,1}\}$$

$$C_3 = \{A_{1,2}, A_{2,1}\}, \quad C_4 = \{A_{1,2}, A_{2,2}\}$$

Each combination of qualitative factors, $C_k, k = 1, ..., 4$, are mapped to,

$$\mathbf{v} = (g_{1,1}\delta_1, g_{1,2}\delta_2, g_{1,3}\delta_3, g_{1,4}\delta_4) \in \mathbb{R}^4,$$

respectively for each k. The term $\delta_k = 1$ if data point, $\mathbf{x}_i$ belongs to category k, whilst other terms, $\delta_k = 0, j = 1, ..., 4, j \neq k$.

2. Hyper-parameters, σ_n^2 (noise variance), σ_p^2 (process variance), $\phi_i, i = 1, ..., n_d$ (length scales for separable kernels), ϕ (length scale for isotropic kernels) and $g_{1,j}, j = 1, ..., n_c$ all re-parameterized to a form such as, $e^{-m(i)}$, to ensure the parameters are constrained to be positive. These hyper-parameters are contained in $\boldsymbol{\theta}$.

3. Assign an initial value to each hyper-parameter in $\boldsymbol{\theta}$ using samples from Gaussian distributions.

4. Use the mean function, $m(\mathbf{x}_i) = h(\mathbf{x}_i)^T\boldsymbol{\beta}$. One way of having a mean function by category is to use An_c basis functions, where A is the number of basis functions used in the algorithm (per category) and n_c is the number of categories. Basis functions take the form for data point $\mathbf{x}_i$,

$$h_{1+A(j-1)} = v_j m(1), \quad h_{2+A(j-1)} = v_j m(2) ..., h_{Aj} = v_j m(A),$$

$\forall j, \ 1 \leq j \leq n_c$. Here, it is assumed that each category has the same basis functions, where $m_i, i = 1, ..., A$ represents the basis functions to use for each category. The terms v_j could either 0 or 1 (referring back to Step 1.). Prior distribution used for $\boldsymbol{\beta}$ is a Gaussian distribution, $N(\boldsymbol{\beta}|\mathbf{b}, B)$. Hyper-parameters, $\boldsymbol{\beta}$ are analytically integrated out such that,

$$\int GP(h(\mathbf{x})^T\boldsymbol{\beta}, k(\mathbf{x}, \mathbf{x}'))MVN(\boldsymbol{\beta}|\mathbf{b}, B)\, d\boldsymbol{\beta} = GP(h(\mathbf{x})^T\mathbf{b}, k(\mathbf{x}, \mathbf{x}') + h(\mathbf{x})^T B h(\mathbf{x}')). \tag{4.3}$$

Variables, $\boldsymbol{\beta}$ are not included in hyper-parameter vector, $\boldsymbol{\theta}$. Hyper-parameters, such as the length-scales, signal and process variances, and the categorical mappings (which are contained in vector $\boldsymbol{\theta}$), are estimated through MAP (or ML).

5. Use an optimization routine such as, scaled conjugate gradient (SCG) to minimize the negative log posterior distribution. The log of the posterior distribution, $\log p(\boldsymbol{\theta}|X, \mathbf{y})$, is given by,

$$\log p(\boldsymbol{\theta}|X, \mathbf{y}) = \log(p(\mathbf{y}|X, \mathbf{b}, B, \boldsymbol{\theta})) + \log(p(\boldsymbol{\theta})) - \log(p(\mathbf{y}|X)) = ...$$
$$... - \frac{1}{2}(\mathbf{y} - (\mathbf{h}(X))^T\mathbf{b})^T(K + (\mathbf{h}(X)^T)B(\mathbf{h}(X)))^{-1}(\mathbf{y} - \mathbf{h}(X)^T\mathbf{b})...$$
$$... - \frac{1}{2}\log|(K + (\mathbf{h}(X))^T)B(\mathbf{h}(X))| - \frac{Q_{tr}}{2}\log(2\pi) + \log(p(\boldsymbol{\theta})) - \log(p(\mathbf{y}|X)).$$

This in maximized in respect to $\boldsymbol{\theta}$, where $p(\boldsymbol{\theta})$ is the prior distribution over the hyper-parameters, $\boldsymbol{\theta}$. $\mathbf{h}(X)^T$ is an $Q_{tr} \times (An_c)$ matrix where each row represents the evaluation of each basis function for a given data point, $\mathbf{x}_i$.

6. Once having obtained $\hat{\boldsymbol{\theta}}$ which is the (global) maximum of $p(\boldsymbol{\theta}|X, \mathbf{y})$, use $\hat{\boldsymbol{\theta}}$ to transform the test points such that,

$\mathbf{x}_i^* = (x_{i,1}^*, x_{i,2}^*, ..., x_{i,n_d}^*, w_{i,j_1}^*, w_{i,j_2}^*, ..., w_{i,j_D}^*)' \rightarrow \mathbf{x}_i^* = (x_{i,1}^*, x_{i,2}^*, ..., x_{i,n_d}^*, v_1, v_2, ..., v_{n_c})'$,
where vector $\mathbf{v} = (v_1, v_2, ..., v_{n_c}) = (\hat{g_{1,1}}\delta_1, \hat{g_{1,2}}\delta_2, ..., \hat{g_{1,j}}\delta_j, ..., \hat{g_{1,n_c}}\delta_{n_c})$

7. Use remaining hyper-parameters, σ_n^2, σ_p^2, $\phi_i, i = 1, ..., n_d$ (for separable kernels), ϕ (for isotropic kernels) to evaluate the moments of the predictive distribution, $p(y_i^*|\mathbf{y}, X, \mathbf{x}_i^*, \hat{\boldsymbol{\theta}})$. Hence,

$$\mathbf{y}(X^*)|\mathbf{y}(X), X, X^* \sim N(\boldsymbol{\mu}_* + K(X^*, X)K(X, X)^{-1}(\mathbf{y} - \boldsymbol{\mu}), ... \qquad (4.4)$$
$$K(X^*, X^*) - K(X^*, X)K(X, X)^{-1}K(X, X^*))$$

, where the mean and kernel shown in Equation 4.3 would be plugged into Equation 4.4. Note that $\boldsymbol{\mu}$ represents the mean function evaluated at each of the training point in the set X, and $\boldsymbol{\mu}_*$ represents the mean function evaluated at each of the test points, $\mathbf{x}_i^*$.

For various other implementations of the Embedded GP model, please turn to Appendix B.

4.8 Other Algorithms

other packages used

Various other algorithms have been used for experimentation which include block-wise Metropolis Hastings (MH), scaled conjugate gradients (SCG) and gradient checking (grad.m), using the NETLAB package (Nabney 2002). NETLAB's implementation of block-wise MH was slightly modified to include an option of choosing a proposal distribution $q(\boldsymbol{\theta})$ based on a drawn uniform random number, α. Gaussian Process Modeling Toolkit (GPML) (Snelson et al. 2011) was used to fit independent GPs for each category where hyper-parameters, $\boldsymbol{\theta}$ are chosen to ML. The GPML code is publicly available at *http : //www.gaussianprocess.org/gpml/code/matlab/doc/index.html*.

our code in MATLAB

Other algorithms used for experimentation purposes such as the Embedded GP model were coded by Sean Tulloch using MATLAB. This includes our implementation of Independent GPs (with a mean function) when optimizing $\boldsymbol{\theta}$ using MAP.

All algorithms use MATLAB (our implementations of various algorithms, NETLAB and GPML), apart from the Treed GP package (Broderick & Gramacy 2010), which uses the R package.

4.9 Making comparisons between Independent GPs and other GPs handling categorical inputs

comparisons between models

Comparisons between the Embedded GP and Independent GPs per category will be needed in the next chapter. However when training n_c GP models separately, this will generate n_c sets of predictive measures such as $Var(y_i^*|\mathbf{x}_i^*, X, \mathbf{y})$ and $E(y_i^*|\mathbf{x}_i^*, X, \mathbf{y})$, and therefore generate n_c sets of validation scores for each model. To make the Independent GP comparable with other models such as the Embedded GP or Hypersphere GP model, we use the separate independent-GPs predictive co-variance matrix structures, K_{C_i}, for each category and use them to build another predictive co-variance matrix of the form,

independent GP models to form combined mean and kernel structures

$$KC = \begin{pmatrix} K_{C_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & K_{C_2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & K_{C_n} \end{pmatrix} \tag{4.5}$$

where K_{C_i} is an $n_i \times n_i$ predictive co-variance matrix, between all the points which belong to category i. The joint predictive mean vector would have the form,

$$MC = \begin{pmatrix} M_{C_1} \\ M_{C_2} \\ \vdots \\ M_{C_n} \end{pmatrix} \tag{4.6}$$

how independent GP models compared to other models

where M_{C_i} would be a $n_i \times 1$ vector containing mean predictions for all data points which belong to category i. Using data structures from Equations 4.5 and 4.6 in validation scores, such as MSE, will generate a single set of validation measures which then can used to compare against the validation measures of other models such as the Embedded GP or Hypersphere GP.

per category validation scores

However, the Embedded GP, Hypersphere GP and Treed GP-LLM models will have predictive mean vector, M, and co-variance matrix, K, which will be of sizes, $Q_{te} \times 1$ and $Q_{te} \times Q_{te}$ respectively where $Q_{te} = \sum_{i=1}^{n_c} n_i$ and n_i is the number of test data points for category i. In this case the predictive mean and co-variance matrix will be of the following forms,

$$M = \begin{pmatrix} M_{C_1} \\ M_{C_2} \\ \vdots \\ M_{C_n} \end{pmatrix}, \; K = \begin{pmatrix} K_{C_1} & K_{C_1,C_2} & \cdots & K_{C_1,C_r} \\ K_{C_2,C_1} & K_{C_2} & \cdots & K_{C_2,C_r} \\ \vdots & \vdots & \ddots & \vdots \\ K_{C_r,C_1} & K_{C_r,C_2} & \cdots & K_{C_n} \end{pmatrix}$$

another comparison between the multi-category GP models and Independent GP models

where K_{C_i} is a $n_i \times n_i$ matrix containing the co-variances between all test data points in category i, whereas K_{C_i,C_j} matrix is $n_i \times n_j$ which contains the covariances between data points from two different categories i and j. Meanwhile the $n_i \times 1$ vector, M_{C_i}, is the mean function for all test data points in category i. Using the individual kernel (sub)-matrix, K_i, and (sub)-mean vector, M_i, for category i in validation measures, allows for comparison of GP models such as Embedded GP (Hypersphere GP/Treed GP-LLMs) for category i against building an independent GP-model for the same category.

4.10 Summary and Next Chapter

conclusions

In this chapter, we discussed our approach of constructing a GP model which can handle a mixture of continuous and categorical inputs. Two types of categorical variables, *ordinal* and *nominal*, have been considered whilst constructing this GP. Other existing GP emulators, such as the Hypersphere GP model were also reviewed, and compared against our GP model. Various suggestions were made on using the Embedded GP in practice. A general implementation of the Embedded GP is presented after, whilst more specific implementations of the various GP models presented in Appendix B and C. These implementations are used in the next chapter. This chapter concluded with discussing how Independent GPs (per category) can be compared against other GP models, such as the Embedded GP.

next chapter

In the next chapter, various GP models (including the Embedded GP) will be used on a number of data sets and compared against using validation scores.

Chapter 5

Experiments and Results

goals of this chapter

This chapter discusses results obtained using the different GP models (including the Embedded GP) reviewed in Chapters 3 and 4 on various data sets. The data sets explored have been derived from a range of sources. Many have been artificially generated to demonstrate properties of the models, but real data and real simulators have also been considered.

Experimentation considers one qualitative factor with n_c number of categories. Data sets have been created using the various data simulators shown below,

1. Tree Simulator - this simulator has data inputs, $\mathbf{x}_i$ which are 3-dimensional, where x_1 and x_2, are continuous and x_3 is categorical. These data inputs represent fertility, diameter and the species respectively for trees. Types of species for this simulator are Douglas Fir, Big Leaf Maple, Willow and Bitter Cherry which are expressed as species 1, 2, 3, and 4 respectively (Wang 1988). Tree height, f_i, is obtained using,

$$f_i = \begin{cases} 4.5 + \exp^{(6.2-5.3(2000x_2^{-0.24})+0.2\log(x_1))} & \text{if } x_3 = 1 \\ 4.5 + \exp^{(5.0-2.7(2000x_2^{-0.35})+0.002(x_1))} & \text{if } x_3 = 2 \\ 4.5 + \exp^{(2.8-1.9(2000x_2^{-0.4})+0.01(x_1))} & \text{if } x_3 = 3 \\ 4.5 + \exp^{(5.5-4.2(2000x_2^{-0.24})+0.01(x_1))} & \text{if } x_3 = 4, \end{cases}$$

   where the diameter, x_1 is fixed to 50, whilst the fertility, x_2 lies in the interval, $[0, 1]$.

2. Friedmann simulator - this simulator originally had a 11-dimensional data input, $\mathbf{x}_i$ where $x_1, ..., x_{10}$ are continuous co-variates and x_{11} is the categorical co-variate (Broderick & Gramacy 2010). The response, f_i was based on this categorical co-variate has follows,

$$f_i = \begin{cases} 10\sin(\pi x_1 x_2) & \text{if } x_{11} = 1 \\ 20(x_3 - 0.5)^2 & \text{if } x_{11} = 2 \\ 10x_4 + 4x_5 & \text{if } x_{11} = 3 \\ 5x_1 + 10x_2 + 20(x_3 - 0.5)^2 + 10\sin(\pi x_4 x_5) & \text{if } x_{11} = 4, \end{cases}$$

   However, we have modified this such that the original continuous co-variates are linearly dependent, hence, $x_i = x_1 + \frac{i-1}{10}, \forall i \in [1, 10]$, effectively transforming the original inputs for required for the Friedmann model into two, one being categorical and the other continuous. This version of the Friedmann model has been used for experimentation. The response f_i is based on the categorical co-variate, x_2 where,

$$f_i = \begin{cases} 10\sin(\pi x_1(x_1 + \frac{1}{10})) & \text{if } x_2 = 1 \\ 20(x_1 + \frac{2}{10} - 0.5)^2 & \text{if } x_2 = 2 \\ 10(x_1 + \frac{3}{10}) + 4(x_1 + \frac{4}{10}) & \text{if } x_2 = 3 \\ 5x_1 + 10(x_1 + \frac{1}{10}) + 20(x_1 + \frac{2}{10} - 0.5)^2 + 10\sin(\pi(x_1 + \frac{3}{10})(x_1 + \frac{4}{10})) & \text{if } x_2 = 4. \end{cases}$$

3. Flute simulator - this is a stochastic, individual-based simulator which runs in discrete time, where two time steps per simulation day represent day and night (Chao et al. 2010) . The code is publicly available using the URL, *www.cs.unm.edu/ ~ /dlchao/flute/*. We have used sample populations using *one–* files provided with this code. These files consider a single community of 2000 people. Multiple re-runs of this simulator were computed; where each run uses a different value for R_0. This term in the simulator is *transmissibility* of the disease, where it is allowed to vary between 1.5 and 2.04 (in step size 0.01). Other parameters in the simulator remain fixed. Running the Flute simulator, gives output information such as the number of susceptible individuals, for particular age groups such as 0-4, 5-18, 19-29, 30-64, 65+.

We used the flute simulator to generate input and output data. GP models are then used to emulate the cumulative number of susceptible individuals (as a percentage of total individuals) at the end of the simulation (hence the final day), for each particular age group. Inputs used in the GP models are, x_1, which is the *transmissibility* parameter, and x_2 which is an ordinal categorical variable, age and the output, f_i which represents the cumulative number of individuals which have been infected (in total on the final day).

Various GP models used for testing

The following models will be used for experimentation,

- Embedded GP (ordinal and nominal cases),
- Hypersphere GP,
- Independent GP (which will be built for each category),
- Treed GP-LLMs.

scaling schemes used

Responses generated using the Tree and Friedmann simulators have been scaled down (per category) accordingly as,

$$y_i = \frac{f_i - f_{A_{1,v},min}}{f_{A_{1,v},max} - f_{A_{1,v},min}} + N(0, \sigma^2).$$

Responses generated using the Flute simulator have been scaled (per category) such that all rescaled responses satisfy,

$$y_i = \frac{f_i - f_{A_{1,v},min}}{f_{A_{1,v},max} - f_{A_{1,v},min}}$$

why scaling responses may be efficient
better conditioning on kernel, K

where $f_{A_{1,v},min}$ and $f_{A_{1,v},max}$ the lowest and highest responses across all data points' responses, f and f* for category $A_{1,v}$. Independent Gaussian noise, of variance σ^2 has been added to the rescaled responses (where responses, f_i are generated using the Tree and Friedmann simulators). Noise, σ^2 was applied to reduce numerical problems.

convergence of SCG

A non-linear optimization routine, SCG, was used for the purpose of minimizing functions in respect to hyper-parameters, $\boldsymbol{\theta}$. Optimization of the parameters, $\boldsymbol{\theta}$, was stopped based on having an accuracy of 10^{-5} in the parameter values, $\boldsymbol{\theta}$ and the function we are trying to minimize, over two consecutive runs of this optimization routine.

5.1 Experiment 1: Comparisons between the Embedded GP (ordinal) and Independent GP models for Tree Simulator

This experiment involves comparing the Embedded GP model (for the ordinal case) against building independent GPs for each category, using one data set. The number of training and test data points used per category was 20 and 25 respectively. Data inputs were generated using Latin Hypercube designs independently for each category. The tree simulator (along with scaling the responses per category and applying additional Gaussian noise with variance, σ^2 of 0.001) was used to generate the outputs.

kernel used

The Matern covariance kernel (with order $v = \frac{5}{2}$) was used, where the covariance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ is given in Table 5.1.

Table 5.1: Co-variance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ used in each algorithm (for Experiment 1). This table, includes notation, $I(\mathbf{x}_i, \mathbf{x}_j)$ which is an indicator function, where $I(.,.) = 1$ if $\mathbf{x}_i = \mathbf{x}_j$ or zero otherwise.

-	Independent GPs (per category)	Embedded GP
$\text{cov}(\mathbf{x}_i, \mathbf{x}_j)$	$\sigma_p^2\left(1+\frac{\sqrt{5}r}{\phi}+\frac{5r^2}{3\phi^2}\right)\exp^{-\frac{\sqrt{5}r}{\phi}}+\sigma_n^2 I(\mathbf{x}_i,\mathbf{x}_j)$	$\sigma_p^2\left(1+\frac{\sqrt{5}r}{\phi}+\frac{5r^2}{3\phi^2}\right)\exp^{-\frac{\sqrt{5}r}{\phi}}+\sigma_n^2 I(\mathbf{x}_i,\mathbf{x}_j)$
r^2	$(x_{i,1}-x_{j,1})^2$	$(x_{i,1}-x_{j,1})^2+(g_{1,k_1}-g_{1,k_2})^2$

parameterization used;parameters in each model

The fourth category, $A_{1,4}$ which has a one-to-one mapping value of $g_{1,4}$ (to be used in the Embedded GP emulation) is fixed to the value of 0. Hence, the hyper-parameters $\boldsymbol{\theta}$ used in the Embedded GP algorithm are the length scale (ϕ), signal variance (σ_p^2), noise variance (σ_n^2) and each of the categorical mappings, $g_{1,k}, k = 1, 2, 3$ (from categories 1 to 3). Each parameter in $\boldsymbol{\theta}$ was parameterized in the form $\theta_b = m_b^2, b = 1, ..., 6$. Independent GP models (per category) do not use the categorical mappings, and $\boldsymbol{\theta}$ only contained the length scale, (ϕ), noise and signal variances, (σ_n^2, σ_p^2 respectively) for this model.

optimizing parameters in the Embedded GP model

The SCG optimization routine was used first to minimize $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$, where $p(\boldsymbol{\theta}|X, \mathbf{y})$ is given by,

$$p(\boldsymbol{\theta}|X,\mathbf{y}) = \frac{GP(\mathbf{y}|0,K)Ga(\phi|1.1,1)Ga(\sigma_n^2|1.1,1)Ga(\sigma_p^2|1.1,1)(\prod_{i=1}^{3}N(g_{1,i}|0,10))}{p(\mathbf{y}|X)}$$

multi-modality issue

Weak informative proper priors were used on the parameters $\boldsymbol{\theta}$. SCG was used 100 times, where each time, different initial values for $\boldsymbol{\theta}$ (where $m_i \sim N(0,1), i = 1, ..., 6$), to find out whether the negative log posterior surface was multi-modal. If multi-modality was found then the lowest minimum was determined.

different modes; different validation scores

The log posterior distribution was found to be multi-modal. Calculating the predictive distribution under each mode and validating each of these models, gave different levels of accuracy in terms of SMSE and NLPD validation measures, as shown in Tables 5.2 and 5.3.

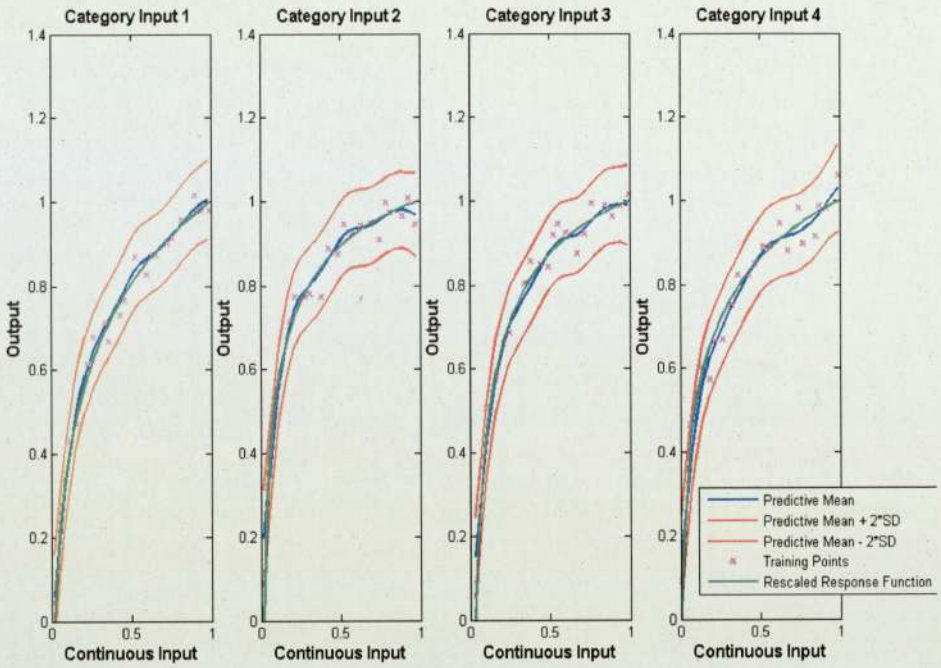


Figure 5.1: Predictive distribution plot for local minimum: -83.3141; where the crosses represent the 20 training points, blue curve is the mean prediction, $E(y_i^*|\mathbf{x}_i^*, D)$ and red curves are the mean predictions minus and plus 2 standard deviations.

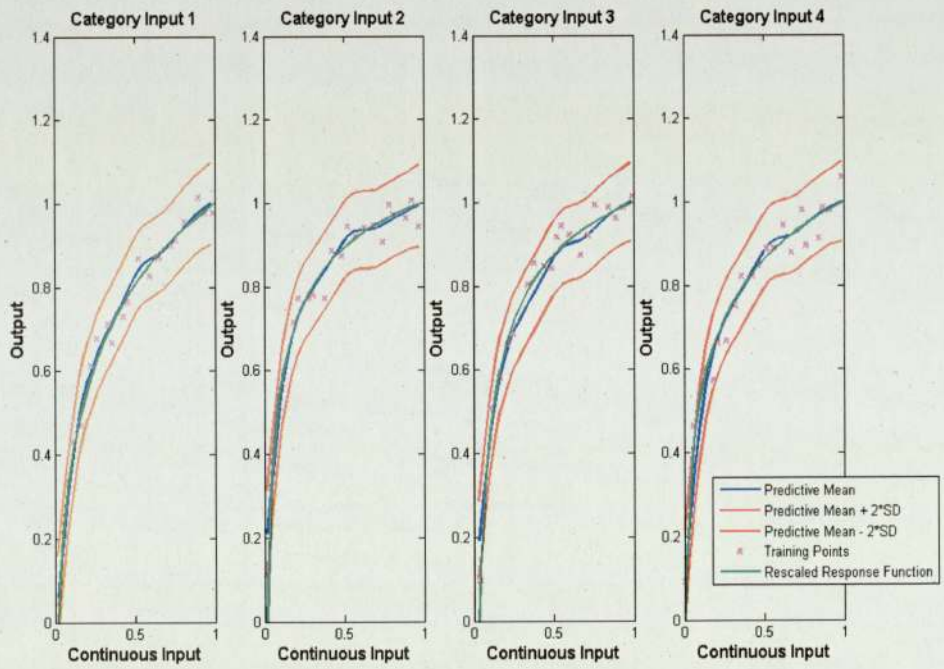


Figure 5.2: Predictive distribution plot for local minimum: -92.9844; where the crosses represent the 20 training points, blue curve is the mean prediction, $E(y_i^*|\mathbf{x}_i^*, D)$ and red curves are the mean predictions minus and plus 2 standard deviations.

Table 5.2: Validation measure NLPD using multiple minimums to evaluate the predictive equations for the Embedded GP model.

$-\log[p(\boldsymbol{\theta}\|X,\mathbf{y})]$	$A_{1,1}$	$A_{1,2}$	$A_{1,3}$	$A_{1,4}$
-77.3730	-1.7921	-1.9557	-1.8838	-1.6105
-81.3614	-1.7427	-1.8218	-1.8568	-1.6742
-82.8527	-1.7622	-1.7594	-1.6960	-1.4913
-83.3141	-1.8194	-1.9268	-1.8422	-1.6533
-85.1811	-1.7377	-1.8731	-1.7536	-1.6290
-87.8311	-1.8378	-1.9406	-1.9128	-1.7426
-88.0612	-1.1818	-1.7886	-1.8945	-1.7192
-88.4568	-1.7514	-1.9128	-1.8838	-1.7250
-92.9844	-1.8224	-1.8440	-1.6496	-1.6445

Table 5.3: Validation measure SMSE using multiple minimums to evaluate the predictive equations for the Embedded GP model.

$-\log[p(\boldsymbol{\theta}\|X,\mathbf{y})]$	$A_{1,1}$	$A_{1,2}$	$A_{1,3}$	$A_{1,4}$
-77.3730	0.0243	0.0211	0.0213	0.0521
-81.3614	0.0271	0.0340	0.0233	0.0472
-82.8527	0.0248	0.0432	0.0355	0.0607
-83.3141	0.0225	0.0261	0.0252	0.0483
-85.1811	0.0272	0.0268	0.0314	0.0480
-87.8311	0.0220	0.0233	0.0198	0.0385
-88.0612	0.0231	0.0383	0.0210	0.0420
-88.4568	0.0265	0.0243	0.0244	0.0426
-92.9844	0.0216	0.0301	0.0387	0.0467

multiple modes; multiple ways to explain the data

Figures 5.1 and 5.2 indicate that this particular data set can be explained by multiple models using the Embedded GP framework. Table 5.4 shows quantile statistics for Dawid score, where each Dawid score was calculated using the predictive equations at each particular minimum.

Table 5.4: Quantile statistics for Dawid score where each Dawid score is calculated using predictive distributions at each particular minimum of $-\log(p(\boldsymbol{\theta}|X,\mathbf{y}))$ using Embedded GP (ordinal) framework.

0.25Q	0.5Q	0.75Q
538.7	540.8	553.3

maximum likelihood to find optimum values

using ML to find the optimum values for hyper-parameters

ML requires the likelihood ($p(\mathbf{y}|X,\boldsymbol{\theta}) = GP(\mathbf{y}|\mathbf{0},K)$), to be maximized. Hence by ML to find the optimal values for $\boldsymbol{\theta}$ using the Embedded GP (ordinal case) framework and each Independent GP (per category) gave the following validation scores shown in Table 5.5 for test points. 100 different initialization for $\boldsymbol{\theta}$ were used to ensure that the (absolute) minimum of $p(\mathbf{y}|X,\boldsymbol{\theta})$ was found, using each GP framework. Predictive plots obtained by using these GP models are shown in Figure 5.3.

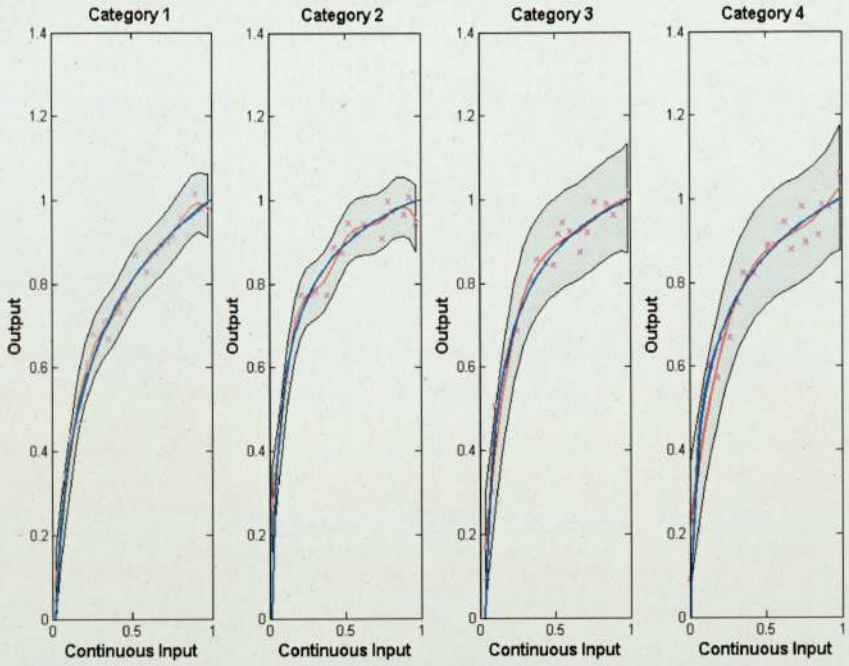
summary of findings

The Embedded GP is doing better in terms of all three validation scores, MSE, NLPD, and Dawid score than using separate independent GPs (per category), when parameters, $\boldsymbol{\theta}$, are optimized through ML. This suggests that for this particular data set is benefiting using the categorical mappings. Using independent GPs (per category) gave a single mode for $p(\mathbf{y}|X,\boldsymbol{\theta})$ across all categories whereas, $p(\mathbf{y}|X,\boldsymbol{\theta})$ was multi-modal using the Embedded GP framework. Table 5.5 also indicates that

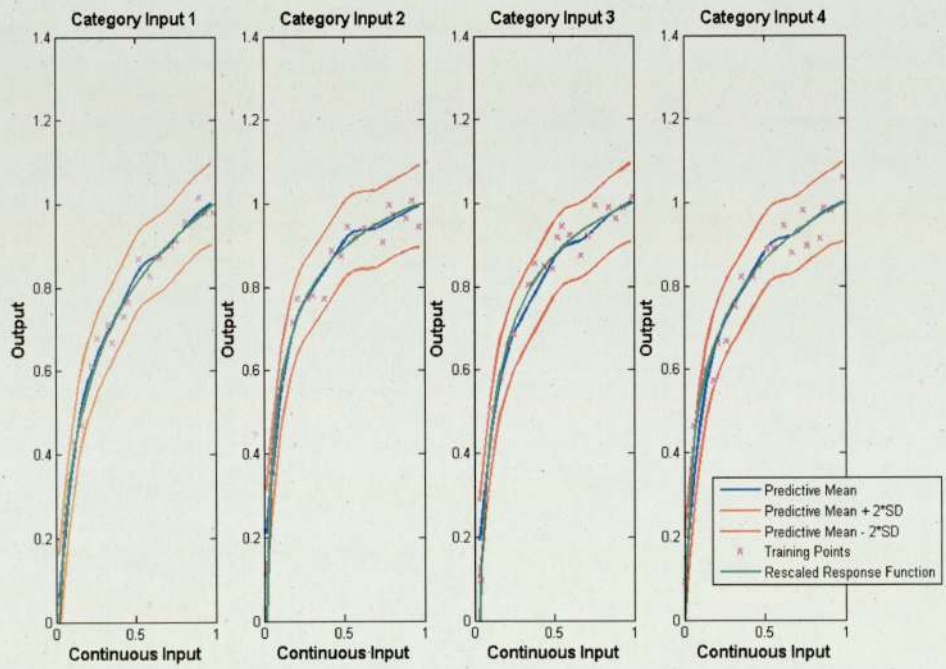
Table 5.5: Contrasts between Embedded GP (ordinal) where hyper-parameters optimized through MAP/ML and Independent GP where hyper-parameters optimized through ML using Tree simulator.

Model-Validation	MSE	NLPD	Dawid score
ML (Embedded GP Ordinal)	0.002	-1.74	538.6
ML (Independent GP per category)	0.003	-1.64	513.4
MAP (Embedded GP Ordinal)	0.002	-1.74	538.7

the prior over $\boldsymbol{\theta}$ has little influence over the predictive quality of the Embedded GP model, and this is because weak informative proper priors were chosen.



(a) Predictive Plot using (lowest) minimum of $-\log(p(\mathbf{y}|X,\boldsymbol{\theta}))$ (independent Gaussian processes (GP) category).



(b) Predictive Plot using (lowest) minimum of $-\log(p(\mathbf{y}|X,\boldsymbol{\theta}))$ (Embedded GP for ordinal categorical variables).

Figure 5.3: Predictive distributions using Embedded GP (ordinal case) and independent Gaussian processes (GP) (per category) frameworks.

5.2 Experiment 2: Comparisons between using MAP and Bayesian Inference for Embedded GP (ordinal) model on Tree Simulator

overview about experiment 1

Experiment 1 used ML and MAP to find plug-in estimates for the hyper-parameters, which were later used in the predictive distribution, $p(y^*|\mathbf{x}^*, D, \boldsymbol{\theta})$. However, predictions were made on one single plug-in multi-variate vector, $\boldsymbol{\theta}$, which is not very Bayesian. This experiment, uses the Embedded GP (for the ordinal case) model with the same data set generated in Experiment 1. The same Matern co-variance kernel used in the Embedded GP (ordinal) framework from Experiment 1 is used. The posterior distribution, $p(\boldsymbol{\theta}|X, \mathbf{y})$ is given by,

computing full Bayesian integration

$$p(\boldsymbol{\theta}|X, \mathbf{y}) = \frac{GP(\mathbf{y}|0, K)Ga(l|1.1, 1)Ga(\sigma_n^2|1.1, 1)Ga(\sigma_p^2|1.1, 1)(\prod_{i=1}^{3} N(g_{1,i}|0, 10))}{p(\mathbf{y}|X)}.$$

Each parameter in $\boldsymbol{\theta}$ was parameterized in the form $\theta_b = m_b^2, b = 1, ..., 6$, where $\theta_1 = \phi$(length scale), $\theta_2 = \sigma_n^2$ (noise variance), $\theta_3 = \sigma_p^2$ (signal variance), and the categorical mappings, $g_{1,i} = \theta_{i+3}$, $i = 4, 5, 6$. Category 4 was fixed to a mapping value of 0.

purpose of this experiment

The purpose of this experiment is to compare the predictions obtained using MAP and Bayesian inference with the Embedded GP framework on the data set generated in Experiment 1. The 'highest' maximum of $p(\boldsymbol{\theta}|X, \mathbf{y})$ with this data set was given by 92.9844 (as found in Experiment 1). Bayesian inference shall now be considered.

Block-wise MH algorithm applied for chosen data set

Firstly, block-wise Metropolis Hastings (MH) is used to sample from the posterior distribution, $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$, over $\boldsymbol{\theta}$. Because multiple modes exist in the posterior surface (as indicated by Experiment 1), the proposal density, $q(\boldsymbol{\theta})$, used for MH is chosen, based on whether a generated uniform random number, U, is less than or equal to some value, α, which was chosen to be 0.2, where,

$$q(\boldsymbol{\theta}) = \begin{cases} (\prod_{i=1}^{3} N(g_{1,i}|0, 0.2))N(\sigma_p^2|0, 0.2)N(\sigma_n^2|0, 0.2) & \text{if } U \leq \alpha \\ \prod_{i=1}^{3} N(g_{1,i}|0, 0.02))N(\sigma_p^2|0, 0.02)N(\sigma_n^2|0, 0.02) & \text{if } U > \alpha \end{cases}$$

The motivation for this proposal density is to allow the algorithm to jump at different rates in the posterior space allowing possibly for it to jump out of local minima. The burn-in period was 10000 samples, and MH algorithm ran for a further 50000 MH-steps (where these samples were retained). Six parallel chains were run. Trace plots of $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$ are shown below in Figure 5.4. Existing code from the Netlab toolbox (metrop.m) was used for this (Nabney 2002).

MH - results being stuck in local minimum

Figure 5.4 demonstrates that each of the chains at some point get stuck in local minima. The parameter space is not being fully explored. Simulated Tempering (ST), with our suggestion (refer to Appendix C), was used as it is able to escape local minima by flattening the target distribution using a series of temperatures on a temperature ladder.

using ST for sampling

ST, requires calculation of the normalization constants, Z_i for each temperature, T_i on the temperature ladder. A temperature ladder of size 3 was used, with the

different temperatures being $1, 1.1, 1.2$. The normalization constants, Z_i, are given by,

$$Z_i = \int \exp^{-(\frac{-\log p(\boldsymbol{\theta})+\log(p(\boldsymbol{\beta}_0))}{T_i})} \, d\boldsymbol{\theta} = \int \frac{\exp^{-(\frac{-\log p(\boldsymbol{\theta})+\log(p(\boldsymbol{\beta}_0))}{T_i})} G(\boldsymbol{\theta})}{G(\boldsymbol{\theta})} \, d\boldsymbol{\theta},$$

which were calculated using an importance sampler, where $\log p(\boldsymbol{\beta}_0) = 92.9844$. 4000 samples were collected using the existing randn function in MATLAB from distribution $G(\boldsymbol{\theta})$, which is a product of Gaussian distributions,

$$G(\boldsymbol{\theta}) = N(\theta_p^2|0, 0.7)N(\theta_n^2|0, 0.01)N(\phi|0, 0.7)\prod_{i=1}^{3}(N(g_{1,i}|0, 0.1)) \tag{5.1}$$

normalization constants

Table 5.6 shows the normalization constants obtained for each of the stationary distributions at these temperatures.

Once normalization constants, Z_i were calculated, four separate parallel runs of ST

Table 5.6: Normalization Constants, Z_i for Temperatures $T_i = 1, 1.1, 1.2$ where $i = 1, 2, 3$ respectively.

T_i	Z_i
1.0	0.245
1.1	0.469
1.2	0.797

ST and Component-wise MH to collect samples

were run, which took just over two weeks. At a given temperature, the component-wise MH algorithm was used to generate samples, where the proposal density $q(\theta_j)$ was chosen based on a generated uniform random number, U, for each component of $\boldsymbol{\theta}$. The choice of proposal density, $q(\theta_j)$ is chosen has follows,

proposal density

$$q(\theta_j) = \begin{cases} N(m_i|0, 0.3) & \text{if } U \leq 0.01 \\ N(m_i|0, 0.2) & \text{if } 0.01 < U \leq 0.5 \\ N(m_i|0, 0.002) & \text{if } U > 0.5 \end{cases}$$

where $\theta_j = m_j^2 \quad j = 1, ..., 6$. Eleven samples of $\boldsymbol{\theta}$ were generated, where each sample was formed using the component-wise MH algorithm, before applying a MH-step to change the temperature. Only samples at the lowest temperature $T_1 = 1$, which sample from the distribution, $\frac{p(\boldsymbol{\theta}|X,\mathbf{y})}{p(\boldsymbol{\beta}_0|X,\mathbf{y})}$ were kept. The burn-in period was 100 temperature MH-iteration steps (which is 6600 component-wise MH steps and 100 MH-steps to the temperature). The first 1924 samples were retrieved from each chain, after the burn-in period. Trace plots for $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$ (over 1924 samples) and each parameter, m_i (for the last 962 samples), are shown in Figures 5.5 and 5.6 respectively.

amount of samples obtained

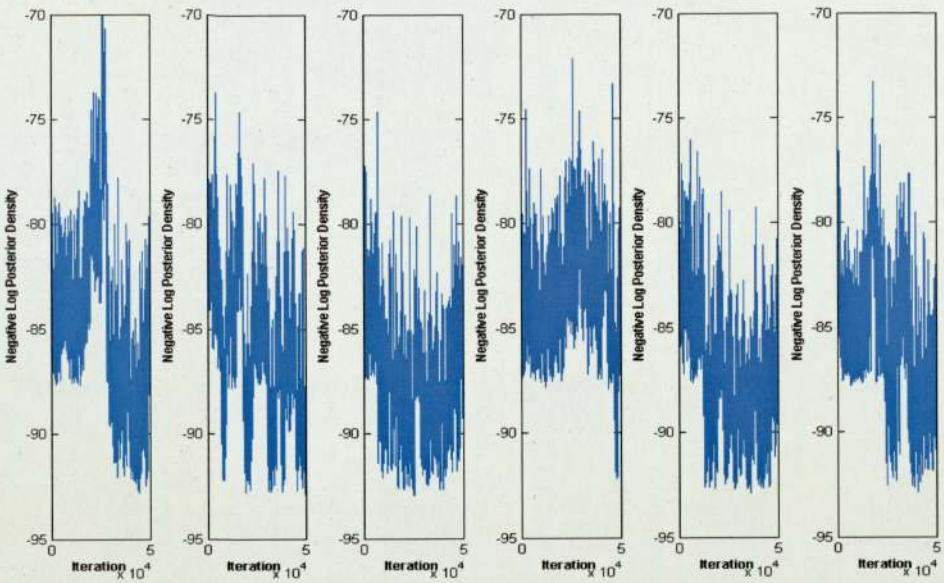


Figure 5.4: Iteration in the MH algorithm vs. -log $p(\boldsymbol{\theta}|X, \mathbf{y})$.

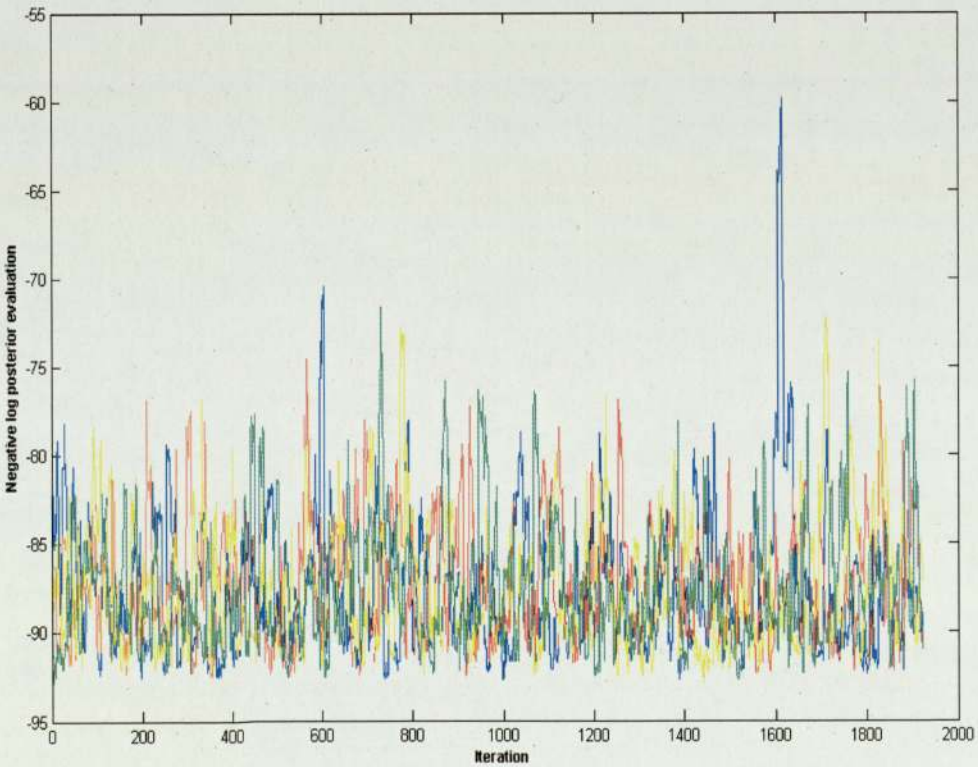
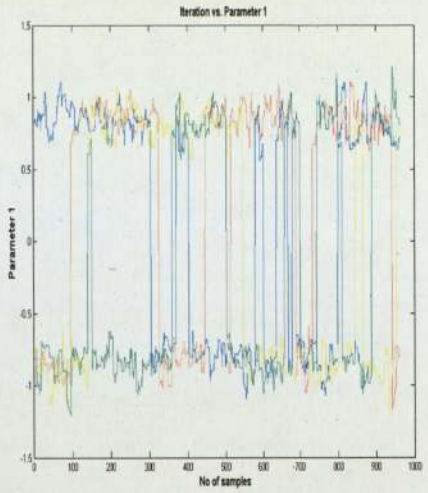
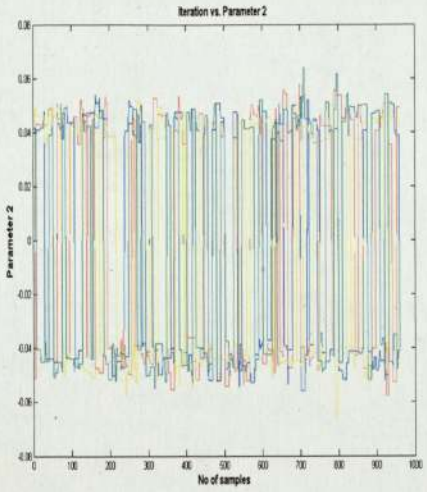


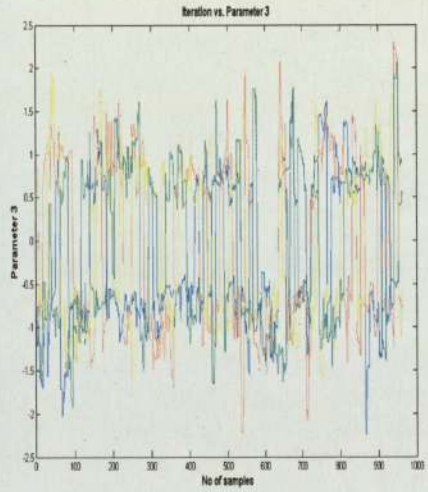
Figure 5.5: $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$ vs. Iteration using Simulated Tempering.



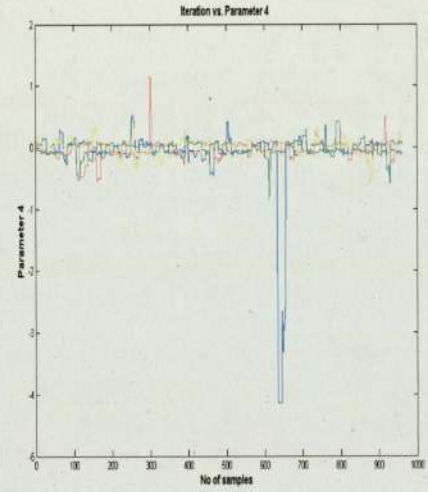
(a) Trace plot for m_1.



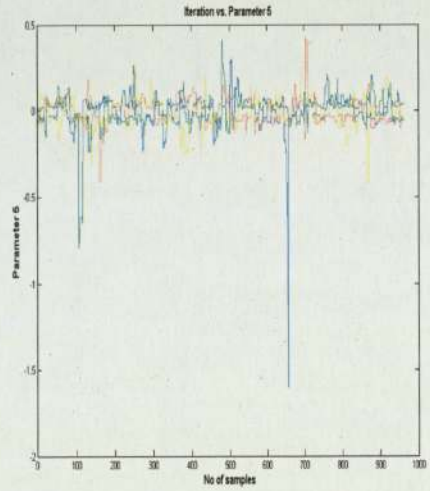
(b) Trace plot for m_2.



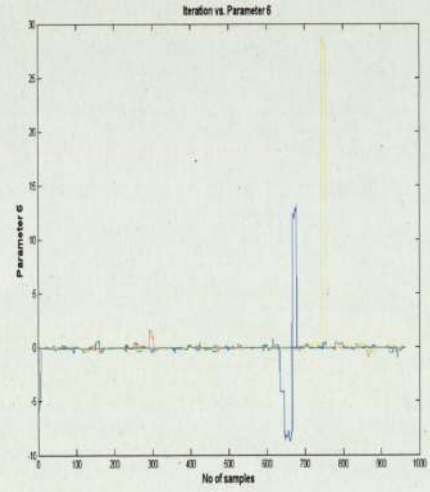
(c) Trace plot for m_3.



(d) Trace plot for m_4.



(e) Trace plot for m_5.



(f) Trace plot for m_6.

Figure 5.6: Trace Plots for parameters $m_1, ..., m_6$ using Simulated Tempering. Variable $m_i = \pm\sqrt{\theta_i}$, where θ_i are the hyper-parameters used in Embedded GP (ordinal) Model. $\theta_1 = \phi$(length scale), $\theta_2 = \sigma_n^2$ (noise variance), $\theta_3 = \sigma_p^2$ (signal variance), and the categorical mappings, $g_{1,i} = \theta_{i+3}$, $i = 4, 5, 6$.

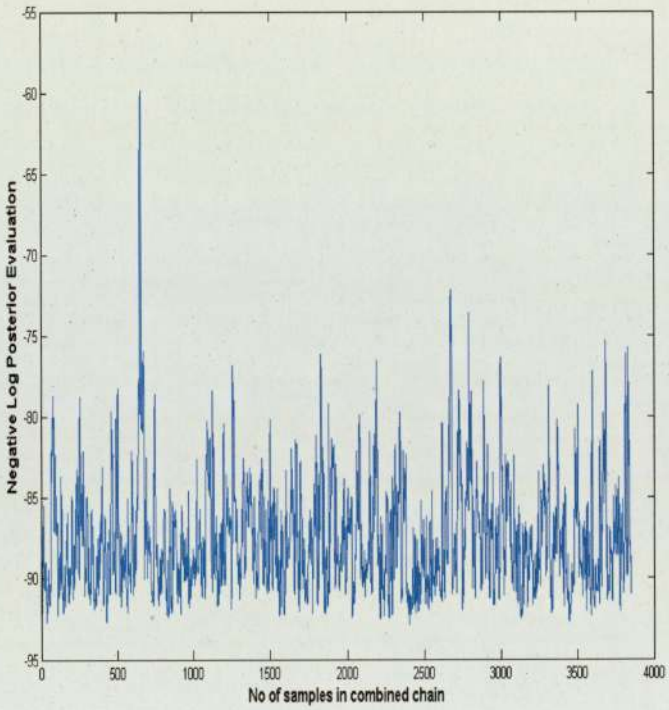


Figure 5.7: -log $p(\boldsymbol{\theta}|X, \mathbf{y})$ vs. Iteration using Simulated Tempering.

Figure 5.9 shows histograms of the posterior distributions, $p(m_i|X, \mathbf{y})$ for each m_i using the final 962 samples from each chain. There appear to be spikes in the trace plots for the parameters, m_4, m_5, m_6. Unlike using MH, ST is able to explore the entire parameter space moving between the different local minima in the surface. All four chains (simulated) have been mixed well, exploring and moving around the various different minima. The trace plot of $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$, after combining the chains together, is shown in Figure 5.7. The samples are combined, by collecting the second half of all samples from the first chain and then second half of all samples from the second chain, and so on. The batch-means approach is used on this combined chain, where averaging the quantity, $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$ over the number of iterations, is shown in Figure 5.8.

spikes

combined chains

combined chain's batch mean and trace plots

The batch-means approach was used on the second half of all chains, where the average of $-\log(p(\boldsymbol{\theta}|X, \mathbf{y})$ was calculated iteratively, as shown in Figure 5.10 where all chains seem to be converging to the same average of -87.5.

Figure 5.11 shows the autocorrelation of $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$, for each of the four chains, using all 1924 samples from each chain.

Autocorrelation

To reduce the correlation between the samples, every 8^{th} sample was taken from each chain, such that the autocorrelation from each chain was 0.27. The reason for picking samples (from each chain) at the autocorrelation level of 0.27 was because of time restrictions. Selecting every 8^{th} sample from each chain (which had original size of 1924 per chain) and combining the chains together lead to 964 samples (where each chain had retained 241 samples). Because the potential scale reduction factor (PSRF) requires an even amount of samples, the decision was taken to remove the final sample from each reduced chain, therefore leaving 960 samples in total. For more information regarding the PSRF (Brooks & Gelman 1998), please refer back to Section 2.6.4.

reduce the dependency of samples

number of samples left **after** reducing autocorrelation

Table 5.7 shows the PSRF obtained for each parameter, θ_i $i = 1, ..., 6$ and MPSRF using the 960 samples (240 samples remaining in each chain). Note that these measures were calculated by discarding the first half of samples from each chain, leaving just 120 samples left in each of those chains. The results obtained by using these convergence diagnostics strengthen the interpretation that the four parallel chains have converged, because each of the individual univariate PSRF for each parameter and MPSRF are less than 1.1. Also, Figure 5.12 shows the PSRF for each parameter, θ_i and MPSRF as the number of samples increase starting from 100 samples in each chain (before discarding the first half). Notice how the MPSRF is an upper bound for the PSRF measures.

calculating the PSRF and MPSRF using remaining samples

Each of the diagnostics, trace plots, inspection of parameters, $m_1, ..., m_6$, calculations of the PSRF for each hyper-parameter, θ_i, and MPSRF, and averaging the quantity, $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$, all indicate convergence. Hence, the last 480 samples (from the remaining 960 samples) were used to calculate the mean and variance of the predictive distribution $p(y_i^*|\mathbf{x}_i^*, X, \mathbf{y})$. The predictive plot is shown in Figure 5.13.

lack of un-convergence

Validation measures such as MSE, NLPD, and Dawid score were used to calculate the performance of the Embedded GP model (for the ordinal case) under MAP and Bayesian integration are shown in Table 5.8. All validation measures in Table 5.8 indicate that by performing Bayesian integration and MAP using the chosen data set

comparisons between MAP and full Bayes result and conclusion

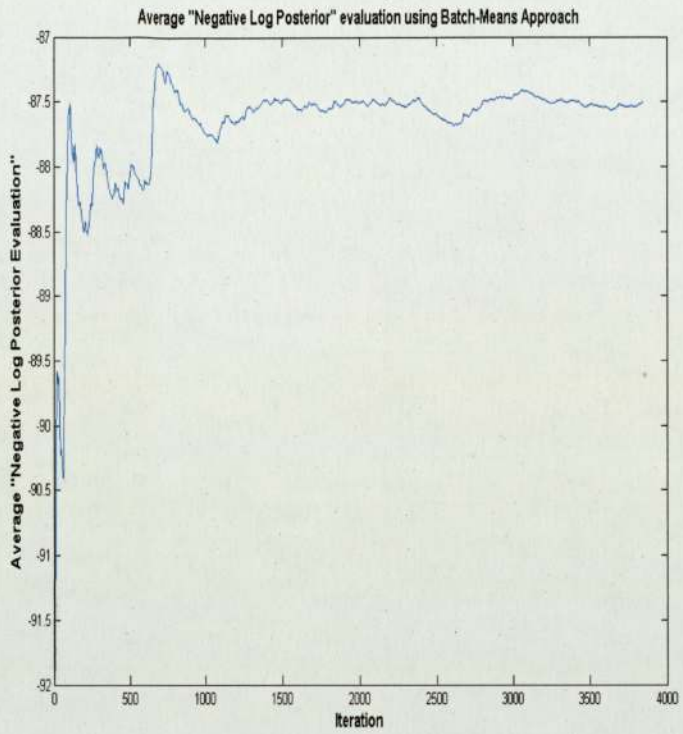
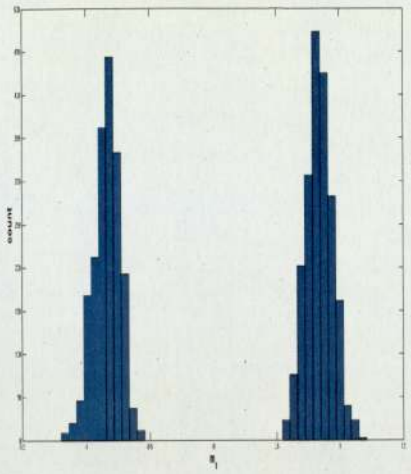


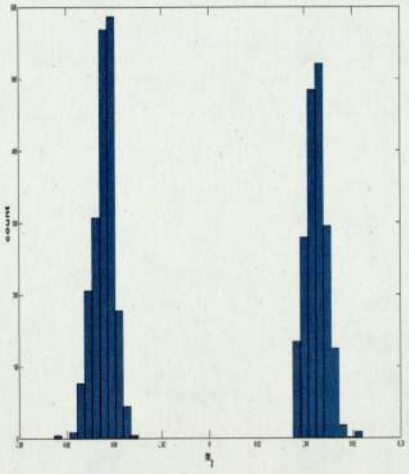
Figure 5.8: $\frac{\sum_{i=1}^{j} -\log p(\boldsymbol{\theta}^{(i)}|X,\mathbf{y})}{j}$ vs. Iteration(j).

Table 5.7: Univariate PSRF per parameter, θ_i and MPSRF.

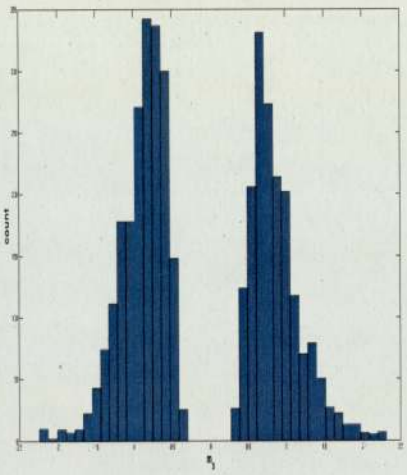
–	PSRF	MPSRF
θ_1	1.0183	-
θ_2	1.0071	-
θ_3	1.0129	-
θ_4	1.0086	-
θ_5	1.0006	-
θ_6	1.0001	-
$\boldsymbol{\theta}$	-	1.0386



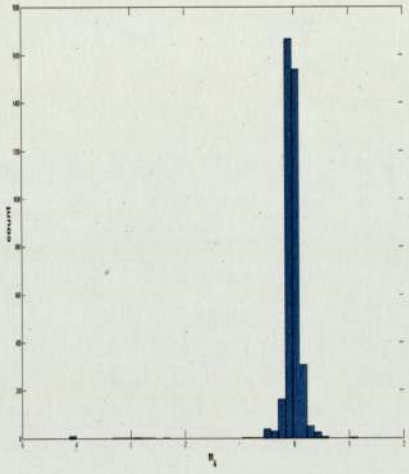
(a) Histogram for m_1.



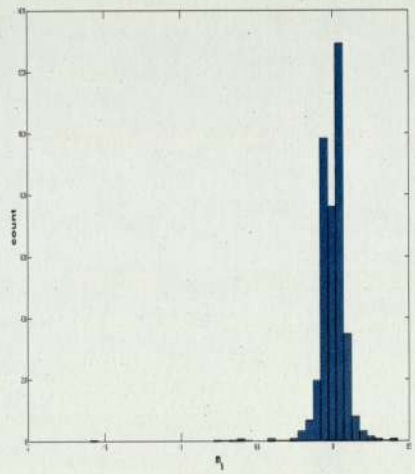
(b) Histogram for m_2.



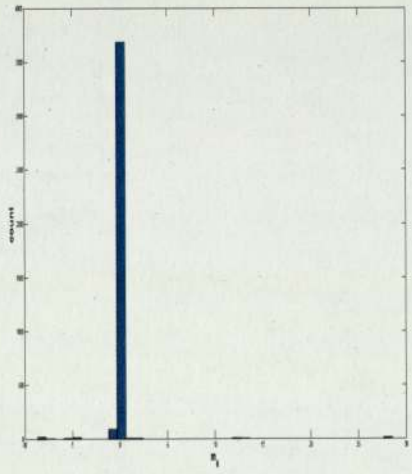
(c) Histogram for m_3.



(d) Histogram for m_4.



(e) Histogram for m_5.



(f) Histogram for m_6.

Figure 5.9: Histograms for all m_i $i = 1, ..., 6$ used in Embedded GP model (using ST). Variable $m_i = \pm\sqrt{\theta_i}$, where θ_i are the hyper-parameters used in Embedded GP (ordinal) Model. $\theta_1 = \phi$(length scale), $\theta_2 = \sigma_n^2$ (noise variance), $\theta_3 = \sigma_p^2$ (signal variance), and the categorical mappings, $g_{1,i} = \theta_{i+3}$, $i = 4, 5, 6$.

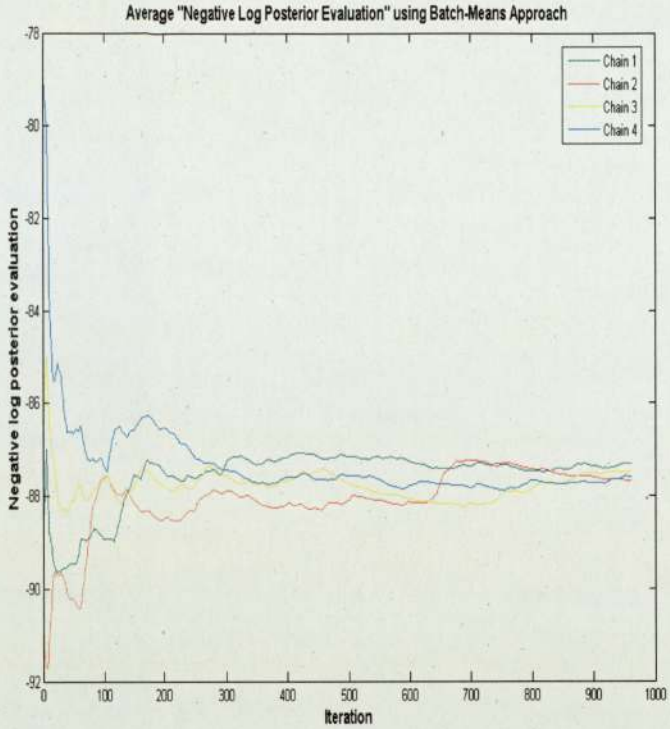
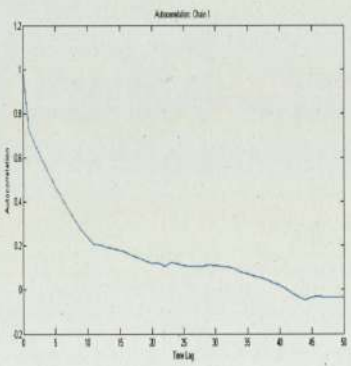
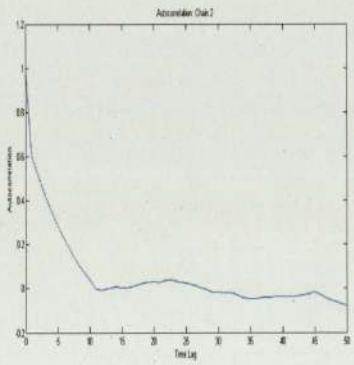


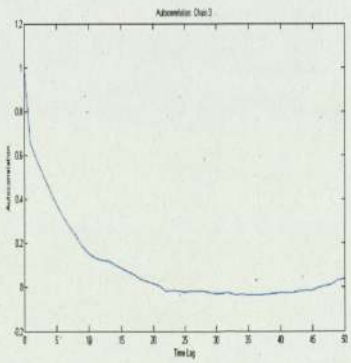
Figure 5.10: $\frac{\sum_{i=1}^{j} -\log p(\boldsymbol{\theta}^{(i)}|X,\mathbf{y})}{j}$ vs. Iteration(j).



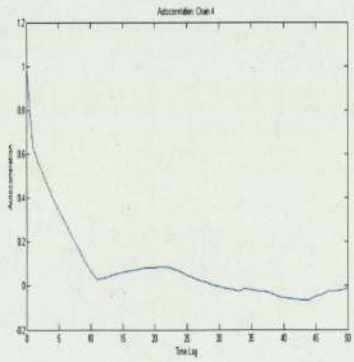
(a) Autocorrelation for Chain 1.



(b) Autocorrelation for Chain 2.



(c) Autocorrelation for Chain 3.



(d) Autocorrelation for Chain 4.

Figure 5.11: Autocorrelations for Chains 1-4. Autocorrelation having a value of zero implies the samples are uncorrelated.

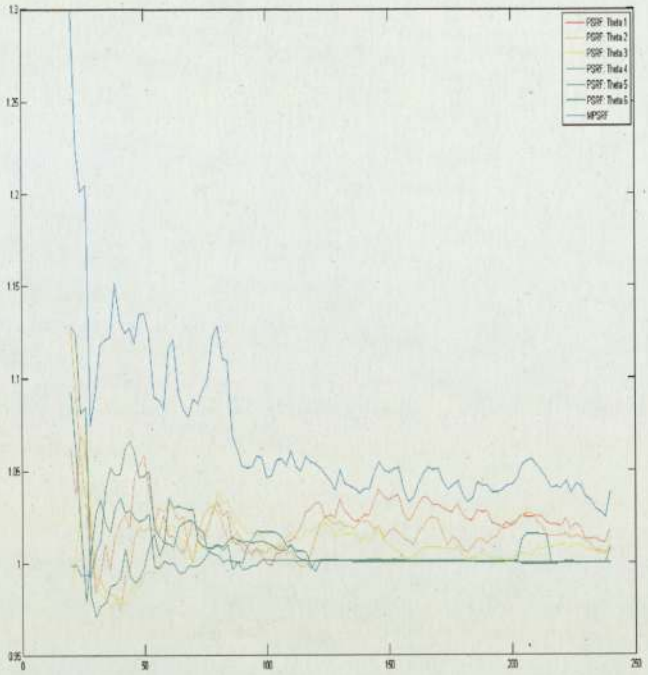


Figure 5.12: MPSRF/ PSRF for each parameter, θ_i $i = 1, ..., 6$ vs. number of samples used in each chain.

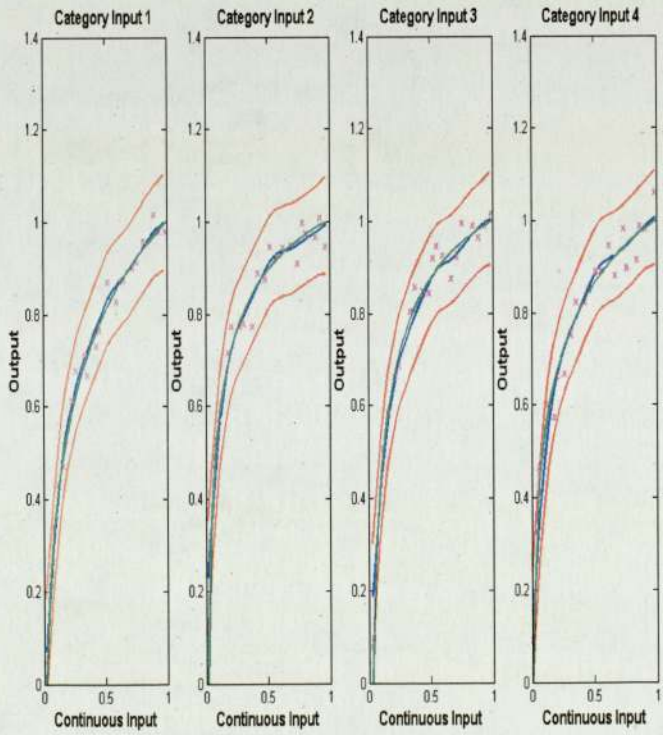


Figure 5.13: Predictive Plot, Using Bayesian integration to integrate out hyper-parameters, $\boldsymbol{\theta}$ in the Embedded GP (ordinal) framework.

Table 5.8: Validation Measures Against Different Models.

Method-Validation	RSE	NLPD	Dawid score
MAP	0.0018	-1.74	538.8
Bayesian Integration (BI)	0.0019	-1.75	545.9

D, there is not a significant improvement offered by integrating out the uncertainty over all hyper-parameters, $\boldsymbol{\theta}$.

5.3 Experiment 3: Comparisons between the Embedded GP (ordinal) and Independent GP models using Friedmann Simulator

purpose

This involves comparing the performance of the Embedded GP (for the ordinal case) against building separate GPs for each category using one data set, where the responses are generated using Friedmann simulator. The number of training and test data points used per category were 20 and 25 respectively. Data inputs were generated using a separate Latin hypercube design per category. The responses were rescaled accordingly, and additional Gaussian noise of variance, $\sigma^2 = 0.001$, was added on to these rescaled responses.

hyper-parameters design

parameterization used

The fourth category, $A_{1,4}$ which has a one-to-one mapping value of $g_{1,4}$ is fixed to the value of 0. The hyper-parameters, $\boldsymbol{\theta}$, used in the Embedded GP framework are the length scale (ϕ), signal variance (σ_p^2), noise variance (σ_n^2) and each of the categorical mappings, $g_{1,k}, k = 1, 2, 3$ (from categories 1 to 3). Each parameter in $\boldsymbol{\theta}$ was parameterized as the form $\theta_b = m_b^2, b = 1, ..., 6$. Independent GP models however do not use the categorical mappings, and $\boldsymbol{\theta}$ only contained the length scale (ϕ), noise and signal variances, (σ_n^2, σ_p^2 respectively) for this model.

choice of kernel used

The Matern kernel (with order $v = \frac{5}{2}$) was used for the GP prior, where the co-variance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ used for each GP model is shown in Tables 5.9

Table 5.9: Co-variance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ used in each model (for Experiment 3).

-	Independent GPs (per category)	Embedded GP (nominal)
$\text{Cov}(\mathbf{x}_i, \mathbf{x}_j)$	$\sigma_p^2\left(1+\frac{\sqrt{5}r}{\phi}+\frac{5r^2}{3\phi^2}\right)\exp^{-\frac{\sqrt{5}r}{\phi}}+\sigma_n^2 I(\mathbf{x}_i,\mathbf{x}_j)$	$\sigma_p^2\left(1+\frac{\sqrt{5}r}{\phi}+\frac{5r^2}{3\phi^2}\right)\exp^{-\frac{\sqrt{5}r}{\phi}}+\sigma_n^2 I(\mathbf{x}_i,\mathbf{x}_j)$
r^2	$(x_{i,1}-x_{j,1})^2$	$(x_{i,1}-x_{j,1})^2+(g_{1,k_1}-g_{1,k_2})^2$

how Embedded GP chooses optimal values for hyper-parameters

The Embedded GP framework was used first, where the optimal values for the hyper-parameters, $\boldsymbol{\theta}$, were chosen such that the negative log posterior distribution, $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$, was minimized. The log posterior distribution is given by,

$$p(\boldsymbol{\theta}|X,\mathbf{y}) = \frac{GP(\mathbf{y}|0,K)Ga(\phi|1.1,1)Ga(\sigma_n^2|1.1,1)Ga(\sigma_p^2|1.1,1)(\prod_{i=1}^{3}N(g_{1,i}|0,10))}{p(\mathbf{y}|X)},$$

different optimization routine used

where weakly informative priors on the hyper-parameters, $\boldsymbol{\theta}$, were chosen. However, because of the multi-modality of the posterior density, $p(\boldsymbol{\theta}|X, \mathbf{y})$, which occurred whilst using the Embedded GP framework in Experiment 1, a different optimization routine, Simulated Annealing (SA), was used to find the optimal $\boldsymbol{\theta}$. This algorithm

was used because it is able to jump out of local modes. The cooling scheme used in the SA algorithm is given by formula,

$$T_k = \frac{T_{k-1}}{1 + T_{k-1}\beta}$$

where $\beta = 0.2$, k represents the time step in SA, and initial temperature, T_1, of 10000 was used. SA was terminated when the system was close to cooling ($T_.$=0.01). Initial parameter values for $\boldsymbol{\theta}$ were chosen randomly, such that each m_i was sampled using a univariate normal distribution with mean 0 and variance 1. SA was run once, where Table 5.10 shows the lowest minimum (of -log $p(\boldsymbol{\theta}|X, \mathbf{y})$), along with its configuration obtained by this routine. Using the SCG optimization algorithm (after using SA)

Table 5.10: Simulated Annealing Results: Embedded GP Framework.

-log($p(\boldsymbol{\theta}\|X, \mathbf{y})$	ϕ	σ_n^2	σ_p^2	$g_{1,1}$	$g_{1,2}$	$g_{1,3}$
-74.1642	0.5277	0.0011	0.5737	-1.1730	-0.4067	-0.5326

combining optimization routines together

with the initial values for the parameters given in Table 5.10 meant SCG converged to a minimum of -74.2105. Minimizing $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$ using (solely just) the optimization routine, SCG, for a 100 different initializations lead to again to finding multiple minima in $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$, where the lowest minimum found had negative log posterior evaluation, $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$ of -74.2105. The parameter values at this minimum are shown in Table 5.11. The differences between using SA and SCG are

Table 5.11: Running Embedded GP: Final hyper-parameter positions.

$-\log p(\boldsymbol{\theta}\|X, \mathbf{y})$	ϕ	σ_n^2	σ_p^2	$g_{1,1}$	$g_{1,2}$	$g_{1,3}$
-74.2105	0.5525	0.0011	0.6284	-1.2358	-0.4477	-0.5703

differences between optimization routines: SA and SCG

very close in terms of $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$ and the parameter estimates. By using SA has meant getting an approximation to the (lowest) minimum in $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$. The optimization routine, SA took approximately 12 hours to run, compared to 2 hours using SCG with a hundred different initializations for $\boldsymbol{\theta}$. Because of the possibility of SA being stuck in a local minima during its routine, due to the sensitivity the cooling scheme and also the running time of this optimization routine, SA will no longer be considered in this thesis.

disadvantages of SA

Table 5.12 shows quantile statistics for the Dawid score, where each Dawid score was calculated using the predictive equations at each particular minimum, using the Embedded GP (ordinal) framework. This table shows even though the posterior

quantile statistics

Table 5.12: Quantile statistics for Dawid score; where each Dawid score is calculated using predictive distributions at each particular minimum of $-\log p(\boldsymbol{\theta}|X, \mathbf{y})$ using Embedded GP (ordinal) framework.

0.25Q	0.5Q	0.75Q
586.16	586.72	586.78

distribution, $p(\boldsymbol{\theta}|X, \mathbf{y})$ is multi-modal using the Embedded GP (for ordinal case) framework on this data set, the model generated by each mode are very similar to each other, according to Dawid score quantile statistics.

ML requires the likelihood, $p(\mathbf{y}|X, \boldsymbol{\theta}) = GP(\mathbf{y}|\mathbf{0}, K)$ is maximized. Using ML

independent GP vs. Embedded GP model using ML to optimize hyper-parameters

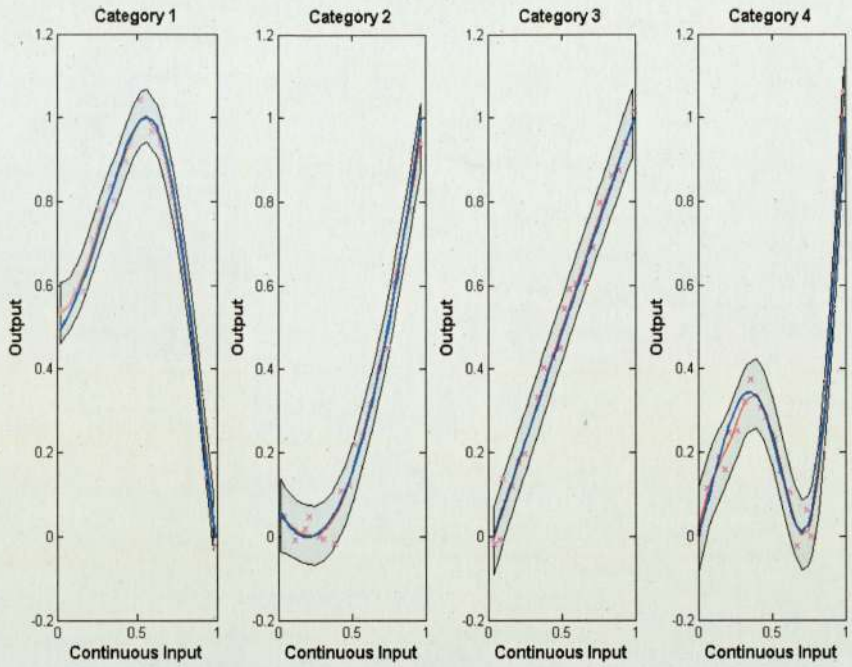
to find the optimal values for hyper-parameters, $\boldsymbol{\theta}$, for both the Embedded GP (ordinal case) and each Independent GP (per category) frameworks gave the following validation scores shown in Table 5.13 on test data points. A hundred different initializations for $\boldsymbol{\theta}$ were used to ensure that the minimum of $-\log p(\mathbf{y}|X, \boldsymbol{\theta})$ was found in both GP models. Predictive plots obtained by using these GP models are shown in Figure 5.14. The Embedded GP (for ordinal case) which used MAP to find the optimal values for $\boldsymbol{\theta}$ also has its validation scores presented in Table 5.13. The Em-

Table 5.13: Contrasts between Embedded GP (ordinal) where hyper-parameters optimized through MAP/ML and Independent GP where hyper-parameters optimized through ML using data where the responses were generated using Friedmann simulator

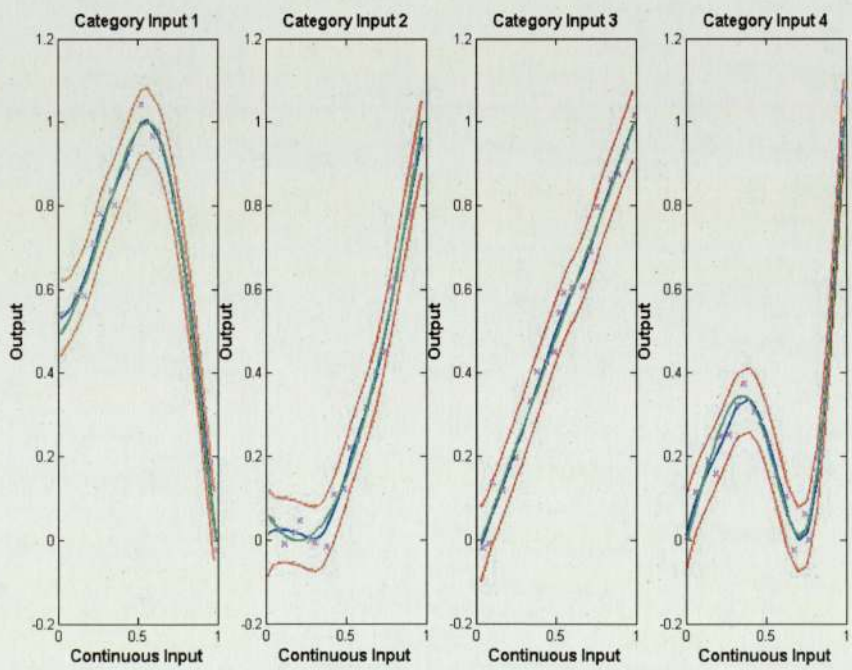
Model-Validation	MSE	NLPD	Dawid score
ML (Embedded GP Ordinal)	0.001	-2.00	586.5
ML (Independent GP per category)	0.001	-1.99	585.6
MAP (Embedded GP Ordinal)	0.001	-1.99	586.1

summary of findings

bedded GP does slightly better in terms of all three validation scores, MSE, NLPD, and Dawid score than using independent GPs (per category). This suggests that for this particular data set (which was generated using the Friedmann simulator), the Embedded GP is not benefiting much from learning the categorical mappings. Using independent GPs (per category) gave a single mode for each individual $p(\mathbf{y}|X, \boldsymbol{\theta})$ across all categories whereas, $p(\mathbf{y}|X, \boldsymbol{\theta})$ was still multi-modal using the Embedded GP (ordinal) framework. Also, Table 5.13 shows that the prior, $\boldsymbol{\theta}$ has little influence over the predictive quality of the Embedded GP model, because weakly informative priors were chosen.



(a) Predictive Plot using (lowest) minimum of $-\log p(\mathbf{y}|X,\boldsymbol{\theta})$ (Independent Gaussian Processes per category).



(b) Predictive Plot using (lowest) minimum of $-\log p(\mathbf{y}|X,\boldsymbol{\theta})$ (Embedded GP (ordinal) model).

Figure 5.14: Predictive distributions using the Embedded GP (ordinal case) and Independent Gaussian Processes per category.

5.4 Experiment 4: Comparisons between Embedded GP Nominal and Ordinal models for Tree Simulator

purpose

Using the same data set generated in Experiment 1, comparisons were made between using the Embedded GP model for the cases of ordinal and nominal categorical variables using validation scores, MSE, NLPD and Dawid score. Category 4 was fixed to a value of 0.004 (having mapping value of, $g_{1,4} = 0.004$). The Embedded GP model for ordinal categorical variables were re-parameterized hyper-parameters contained in $\boldsymbol{\theta}$ as $\theta_i = m_i^2, i = 1, ..., 6$ and for the nominal case, $\theta_i = e^{-m_i}, i = 1, ..., 6$. The posterior distribution

$$p(\boldsymbol{\theta}|X, \mathbf{y}) = \frac{GP(\mathbf{y}|\mathbf{0}, K)Ga(\phi|1.1, 1)Ga(\sigma_n^2|1.1, 1)Ga(\sigma_p^2|1.1, 1)(\prod_{i=1}^{3} N(g_{1,i}|0, 10))}{p(\mathbf{y}|X)},$$

finding optimal solutions for hyper-parameters

was maximized in order to find the optimal values for the hyper-parameters, $\boldsymbol{\theta}$. The SCG optimization algorithm, was used to compute this task, using 100 different initializations for $\boldsymbol{\theta}$, where each m_i was initialized using a univariate Gaussian distribution, $N(0, 1)$. The Matern kernel (with order $v = \frac{5}{2}$) was used in the GP prior for the Embedded GP, where the particular forms of the kernel are shown in Table 5.14. Results obtained from calculating MSE, NLPD and Dawid Score are shown

Table 5.14: Form of Kernels Used.

-	Embedded GP (Ordinal)	Embedded GP (Nominal)
$\text{Cov}(\mathbf{x}_i, \mathbf{x}_j)$	$\sigma_p^2\left(1+\frac{\sqrt{5}r}{\phi}+\frac{5r^2}{3\phi^2}\right)\exp^{-\frac{\sqrt{5}r}{\phi}}+\sigma_n^2 I(\mathbf{x}_i, \mathbf{x}_j)$	$\sigma_p^2\left(1+\frac{\sqrt{5}r}{\phi}+\frac{5r^2}{3\phi^2}\right)\exp^{-\frac{\sqrt{5}r}{\phi}}+\sigma_n^2 I(\mathbf{x}_i, \mathbf{x}_j)$
r^2	$(x_{i,1} - x_{j,1})^2 + g_{1,k_1}^2 + g_{1,k_2}^2$	$(x_{i,1} - x_{j,1})^2 + (g_{1,k_1} - g_{1,k_2})^2$

findings and summary

in Table 5.15. This shows there is a further improvement, when using the Embedded GP assuming the qualitative categorical variable, *species* is nominal, for this particular data set. Unlike using the Embedded GP (ordinal) framework, maximizing the posterior distribution $p(\boldsymbol{\theta}|X, \mathbf{y})$ in the Embedded GP (nominal) framework, using optimization routine, SCG, only gave one unique single mode.

Table 5.15: Comparisons between the Embedded GP model for ordinal and nominal categorical variables.

-	MSE	NLPD	Dawid score
Embedded GP (Nominal categorical variables))	0.002	-1.84	559.4
Joint-GP (Ordinal categorical variables)	0.002	-1.74	538.7

5.5 Experiment 5: Comparisons between various GP models using Multiple Data Sets

purpose; conclusions from experiments 1 and 3

This experiment involves using the Embedded GP (nominal) framework on a number of different training and test data sets. The Embedded GP (ordinal) model used in Experiment 1 performs significantly better (in terms of RSE, NLPD, and Dawid score) on the generated data set than using Independent GPs (per category). Results obtained in Experiment 3 indicate there is very little benefit of learning the categorical mappings. The purpose of this experiment, is to ensure that these findings are consistent with other generated data sets. Other GP models, such as the Hypersphere GP and Treed GP (with Limiting linear models (LLM)) were also used in this experiment. Treed GP-LLMs were used with the default settings, with options basemax= 1 and splitmin= 2 being used.

The Matern covariance kernel (with order $v = \frac{5}{2}$) is used by each of the GP models. The covariance between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$, used by each of the GP models, is given in Tables 5.16 and 5.17. This excludes the Treed GP-LLM model. form of kernel used

Table 5.16: Kernel function used in Experiment 5 for Embedded GP and Hypersphere GP Models

-	Embedded GP (nominal)	Hypersphere GP model
$\text{Cov}(\mathbf{x}_i, \mathbf{x}_j)$	$\sigma_p^2\left(1+\sqrt{5}r+\frac{5r^2}{3}\right)\exp^{-\sqrt{5}r}+U$	$\sigma_p^2 T(A_{1,k_1}, A_{1,k_2})\left(1+\frac{\sqrt{5}r}{\phi}+\frac{5r^2}{3\phi^2}\right)\exp^{-\frac{\sqrt{5}r}{\phi}}+U$
r^2	$\frac{1}{\phi_1^2}(x_{i,1}-x_{j,1})^2+g_{1,k_1}^2+g_{1,k_2}^2$	$(x_{i,1}-x_{j,1})^2)$

Table 5.17: Kernel function used in Experiment 5 for Independent GP Model

-	Independent GP
$\text{Cov}(\mathbf{x}_i, \mathbf{x}_j)$	$\sigma_p^2\left(1+\frac{\sqrt{5}r}{\phi}+\frac{5r^2}{3\phi^2}\right)\exp^{-\frac{\sqrt{5}r}{\phi}}+U$
r^2	$(x_{i,1}-x_{j,1})^2)$

Variable U which appears in Table 5.16 equates to $\sigma_n^2 I(\mathbf{x}_i, \mathbf{x}_j)$. Separate mean functions per category were used in the GP prior, $p(\mathbf{y}|X, \boldsymbol{\theta}) = GP(m(.), k(.,.))$ under each of the GP frameworks, Hypersphere GP, Embedded GP (nominal) and Independent GPs (per category). Data points belonging to category i use the mean function of the following form, $\beta_{2i} + \beta_{2i-1}x_{i,1}$, where for $j = 1, ..., 2i-2$ and $j = 2i+1, ..., 2n_c$, ther terms β_j are multiplied by basis functions of 0. This is for the Embedded GP and Hypersphere GP frameworks. Independent GPs (per category) use the mean function of $\beta_1 + \beta_2 x_{i,1}$. The parameters, $\boldsymbol{\beta}$ is $2n_c \times 1$-dimensional for the Embedded GP and Hypersphere GP models, and 2×1-dimensional for independent GPs (per category), have been analytically integrated out, where each β_i has a normal prior of N(0,100). Please refer to Appendix B and Section 2.5 for more details. Hyper-parameters, $\boldsymbol{\theta}$ is chosen such that it MAP, where $p(\boldsymbol{\theta}|X, \mathbf{y})$, separate mean functions per category

$$p(\boldsymbol{\theta}|X,\mathbf{y}) = \frac{(\int GP(\mathbf{y}|m,K)\prod_{i=1}^{2n_c} p(\beta_i|0,100)\ d\boldsymbol{\beta})}{p(\mathbf{y}|X)} \times ...$$

$$... \times \frac{Ga(\phi|1.1,1)Ga(\sigma_n^2|1.1,1)Ga(\sigma_p^2|1.1,1)(\prod_{i=1}^{3} Ga(g_{1,i}|1.1,1))}{p(\mathbf{y}|X)}$$

Treed GP (with the Limiting Linear Models (LLM)) is re-run 100 times, and the tree which maximizes the posterior distribution is chosen. The hyper-parameters, $\boldsymbol{\theta}$ used in the Embedded GP (nominal) model are the length scale (ϕ), signal variance (σ_p^2), noise variance (σ_n^2) and each of the categorical mappings, $g_{1,k}, k = 1, 2, 3, 4$ (from categories 1 to 4). Each parameter in $\boldsymbol{\theta}$ was parameterized as the form $\theta_b = e^{-m(b)}, b = 1, ..., 7$. The Hypersphere GP model, however contains the cross-pair parameters, $\theta_{i,j}$, $\forall i < j$ which are used to construct the correlation matrix, T. This GP model also uses the length scale (ϕ), signal and noise variances, σ_p^2 and σ_n^2 respectively, which are contained in $\boldsymbol{\theta}$, which use same re-parameterization for its parameters as the Embedded GP model. Independent GP models however do not use the categorical mappings, and therefore $\boldsymbol{\theta}$ only contains the length scale (ϕ), noise and signal variances, (σ_n^2, σ_p^2 respectively) for this model, where parameters are re-parameterized in the same manner as the other GP models. forms for the hyper-parameters used in each GP model

The change of Gaussian to Gamma priors, $p(\boldsymbol{\theta})$, on the categorical mappings changing the prior

$g_{1,i}, i = 1, ..., 4$ was because these parameters are already constrained to be greater than 0 (through their re-parameterizations). Gamma distributions are distributions over variables, x, where $x \geq 0$, so it made more sense to use type of prior than using normal distributions.

Running SCG multiple times

SCG was run with 100 different initial values for $\boldsymbol{\theta}$ where each m_i was generated using univariate normal distributions, $N(0, 1)$ in order to find optimal values for $\boldsymbol{\theta}$, which maximize $p(\boldsymbol{\theta}|X, \mathbf{y})$. This was done using each GP framework.

generating the data sets

Six independent data sets were generated (labeled as data sets 1-6) using Latin Hypersphere designs (per category) to create the inputs, and the responses, $f_i, i = 1, ..., n$, were generated using the Friedmann simulator. Various noise levels were applied onto the scaled output. Table 5.18 shows the amount of training and test points used for each data set, along with the amount of noise applied σ^2 on to the scaled responses, f_i. Validation scores, such as MSE, Dawid score, NLPD obtained from

Table 5.18: Amount of training and test data points used per category along with the noise variance, σ^2 (Using Tree simulator).

Data Set	No. of training points (per cat.)	No. of test points (per cat.)	σ^2
1-5	30	25	0.01
6	20	25	0.001

performance of each GP model

using the various GP frameworks are shown in Tables 5.19 and 5.20, where the best validation scores (for a given data set) are in bold.

Treed GP-LLMs partition on the given data set correctly on all six data sets

Table 5.19: Validation measures obtained using the Embedded GP (nominal) and Independent GPs (per category) frameworks for each training and test data set (Using Friedmann simulator).

Embedded GP (nominal)				Independent GP per cat.			
Data Set	MSE	NLPD	Dawid score	Data Set	MSE	NLPD	Dawid score
1	**0.0114**	**-0.82**	346.7	1	0.0117	-0.80	340.8
2	**0.0102**	**-0.88**	**355.4**	2	0.0103	-0.84	350.9
3	0.0172	-0.57	**306.8**	3	**0.0165**	**-0.57**	304.9
4	0.0157	-0.55	300.5	4	**0.0155**	**-0.56**	**302.7**
5	**0.0110**	**-0.83**	349.6	5	**0.0110**	-0.83	**350.3**
6	0.0012	-1.93	581.5	6	**0.0011**	**-1.99**	**588.7**

Table 5.20: Validation measures obtained using the Treed GP-LLM and Hypersphere GP models on each training and test data set (Using Friedmann simulator).

Treed GP-LLM model					Hypersphere GP			
Data Set	MSE	NLPD	Dawid score	No of Partitions	Data Set	MSE	NLPD	Dawid score
1	0.0115	-0.81	**347.8**	4	1	0.0116	-0.81	343.2
2	0.0105	-0.85	352.5	4	2	**0.0102**	-0.86	351.2
3	0.0186	-0.52	278.0	4	3	0.0172	-0.55	304.8
4	0.0159	**-0.59**	300.1	4	4	0.0154	-0.55	300.7
5	**0.0110**	-0.81	349.6	4	5	0.0114	-0.82	346.1
6	0.0012	-1.94	569.2	4	6	0.0013	-1.87	564.1

summary and conclusions from using data sets 1-6

and the validation results from this model indicates this GP model is able to give just as good predictions as the other GP models which are already considerably very close (in terms of validation). Using the six generated data sets, for a hundred re-runs of using the Treed GP-LLM framework, there were no other forms of partitioning, other than a separate partition for each category. Hypersphere GP and Embedded GP models both gave very close validation scores especially with data set

4. According to data sets 1-6, there is little benefit from learning the categorical mappings in terms of using the Embedded GP (and Hypersphere GP) frameworks respectively, because validation scores indicate that both GP models give as good as predictive performance as using Independent GPs per category. For data set 6 there is a slight difference between the Embedded GP, Independent-GP per category, Treed GP-LLMs, and Hypersphere GP models across all validation scores.

Further data sets

Seven further data sets were generated with the change of simulating the responses using Tree simulator rather than Friedmann simulator. Apart from that, all conditions imposed earlier on data sets 1-6 were used. Table 5.21 shows the amount of training and test points used along with the amount of noise applied, σ^2, on to the scaled output used for data sets 7-13.

conditions imposed by for data sets 7-13

validation scores

Validation scores from using various GP models are shown in Tables 5.22 and 5.23, where the best validation scores are in bold.

summary and conclusions from using data sets 7-13

Using data sets 7-13 in the Treed GP-LLM model has meant this model struggled to partition correctly on the given data set, often placing responses from different categories under the same leaf node. This is because the responses generated using the Tree simulator for different categories (which are scaled down per category) are very similar. However, there are instances when the Treed GP-LLM could benefit from this, for example, for data set 7. Unlike Data sets 1-6, other partitions (and predictive distributions) were obtained on multiple re-runs of the Treed GP-LLMs using data sets 7-13. However these were discarded because they were not the highest posterior probability obtained. Both the Embedded GP (nominal) and the Hypersphere GP models always perform better for all validation scores than using Independent GPs (per category). Validation scores between the Embedded (nominal) GP and Hypersphere GP are always very close, in terms of all validation scores, for most data sets, apart from data set 13.

correlation matrix obtained using Hypersphere GP model

Figures 5.15 and 5.16 shows the correlation matrix, T (obtained after selecting values for hyper-parameters, $\boldsymbol{\theta}$ which MAP), between the categories for data sets 1, 3, 7 and 8. These matrices confirm (very) strong correlations for data sets 7 and 8 as all the responses from each category are similar whereas data sets 1 and 3 show that the Hypersphere GP model is able to capture the negative correlations between categories, whereas the Embedded GP (nominal) is constrained.

Predictive distributions using data sets 1, 2, 7 and 13 are shown in Figures 5.17 - 5.20 across all GP models, apart from the Treed GP-LLM model. Predictive plots obtained from using the Treed-LLM framewor for data sets 1 and 13 are shown in Figures 5.22 and 5.21.

Table 5.21: Amount of training and test data points per category along with the noise variance, σ^2 (Using Tree simulator).

Data set	No. of training points (per cat.)	No. of test points (per cat.)	σ^2
7-11	30	25	0.01
12-13	20	25	0.001

Table 5.22: Validation measures obtained using Independent GPs per category and Embedded GP (nominal) model for each training and test data set (Using Tree simulator).

Embedded GP (nominal)				Independent GP per cat.			
Exp	MSE	NLPD	Dawid score	Exp	MSE	NLPD	Dawid score
7	**0.0101**	**-0.88**	**368.5**	7	0.0132	-0.77	347.0
8	**0.0102**	**-0.87**	**359.3**	8	0.0118	-0.78	339.4
9	**0.0124**	**-0.76**	**336.6**	9	0.0130	-0.69	321.8
10	**0.0125**	-0.76	343.8	10	0.0140	-0.66	322.0
11	0.0110	-0.83	350.3	11	0.0113	-0.81	345.0
12	**0.0022**	**-1.77**	**538.9**	12	0.0028	-1.43	467.9
13	**0.0029**	**-1.46**	**490.3**	13	0.0041	-1.06	379.8

Table 5.23: Validation measures obtained using the Treed GP-LLM and Hypersphere GP model for data sets 8-13 (Using Tree simulator).

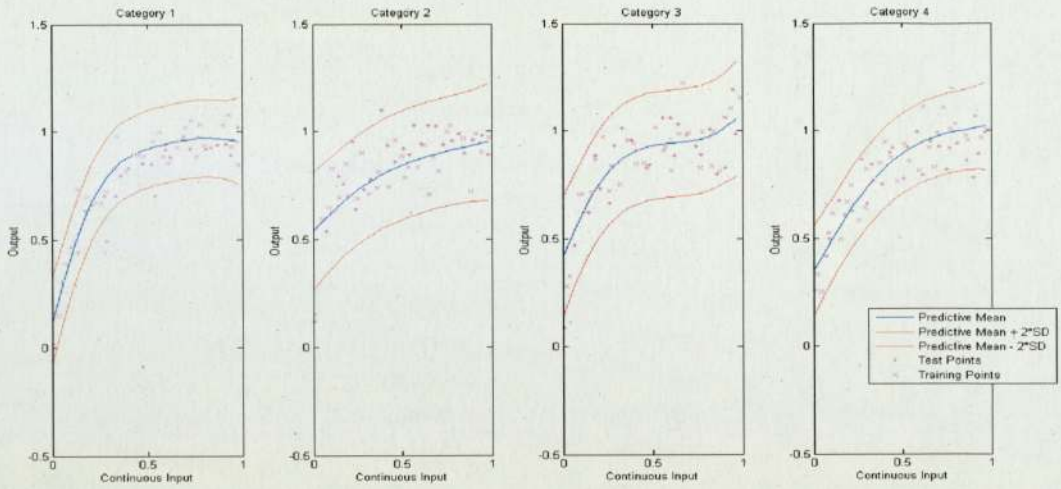
Treed GP-LLM model					Hypersphere GP			
Data set	MSE	NLPD	Dawid score	Amount of Partitions	Exp	MSE	NLPD	Dawid score
7	0.0098	-0.89	361.7	1	7	0.0103	-0.87	367.7
8	0.0127	-0.76	340.0	2	8	**0.0102**	-0.87	358.8
9	0.0160	-0.65	312.8	2	9	0.0125	-0.77	336.6
10	0.0118	-0.79	339.7	2	10	**0.0125**	**-0.77**	**345.0**
11	0.0118	-0.80	343.7	1	11	**0.0109**	**-0.84**	**351.0**
12	0.0023	-1.58	497.5	2	12	0.0028	-1.72	526.3
13	0.0050	-1.24	416.8	3	13	0.0031	-1.45	483.5

$$\begin{pmatrix} 1.0000 & -0.8192 & 0.3916 & -0.4250 \\ -0.8192 & 1.0000 & -0.8402 & 0.0751 \\ 0.3916 & -0.8402 & 1.0000 & 0.1504 \\ -0.4250 & 0.0751 & 0.1504 & 1.0000 \end{pmatrix}, \begin{pmatrix} 1.0000 & 0.9505 & 0.9737 & 0.9754 \\ 0.9505 & 1.0000 & 0.9934 & 0.9923 \\ 0.9737 & 0.9934 & 1.0000 & 0.9922 \\ 0.9754 & 0.9923 & 0.9922 & 1.0000 \end{pmatrix}$$

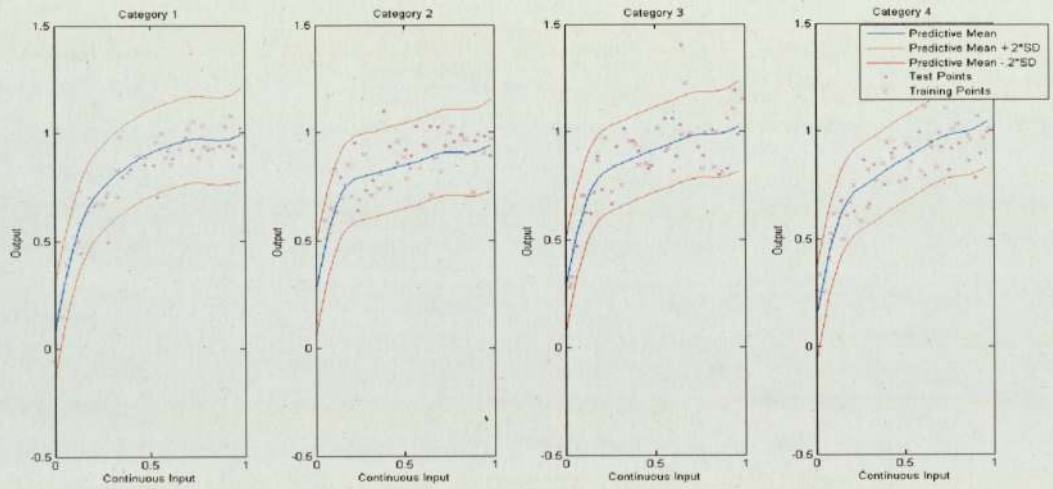
Figure 5.15: T matrix obtained using Data sets 1 and 7 respectively.

$$\begin{pmatrix} 1.0000 & -0.5019 & 0.2510 & -0.5352 \\ -0.5019 & 1.0000 & -0.9548 & 0.1071 \\ 0.2510 & -0.9548 & 1.0000 & -0.0579 \\ -0.5352 & 0.1071 & -0.0579 & 1.0000 \end{pmatrix}, \begin{pmatrix} 1.0000 & 0.9864 & 0.9869 & 0.9847 \\ 0.9864 & 1.0000 & 0.9984 & 0.9983 \\ 0.9869 & 0.9984 & 1.0000 & 0.9994 \\ 0.9847 & 0.9983 & 0.9994 & 1.0000 \end{pmatrix}$$

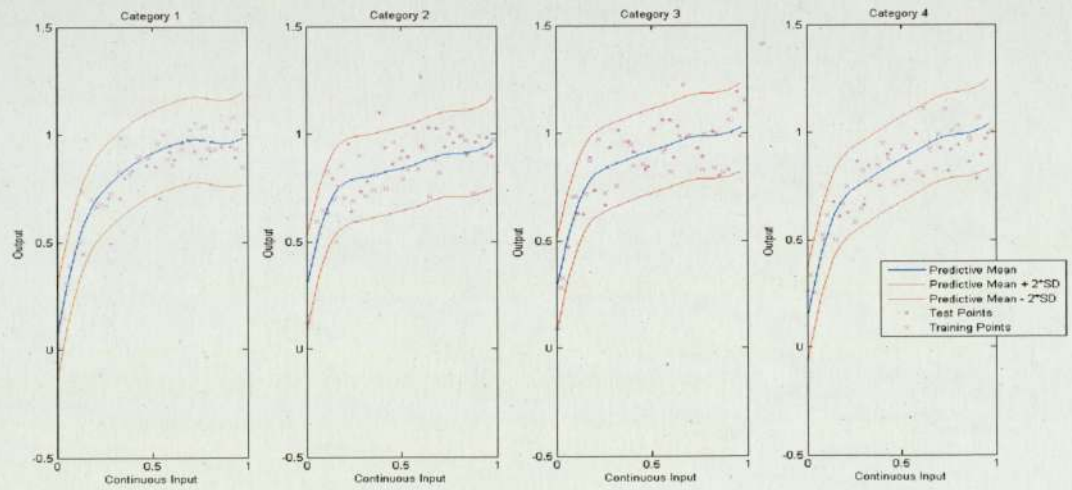
Figure 5.16: T matrix obtained using data sets 3 and 8 respectively.



(a) Independent GP Predictive Plot per category for Data Set 7.

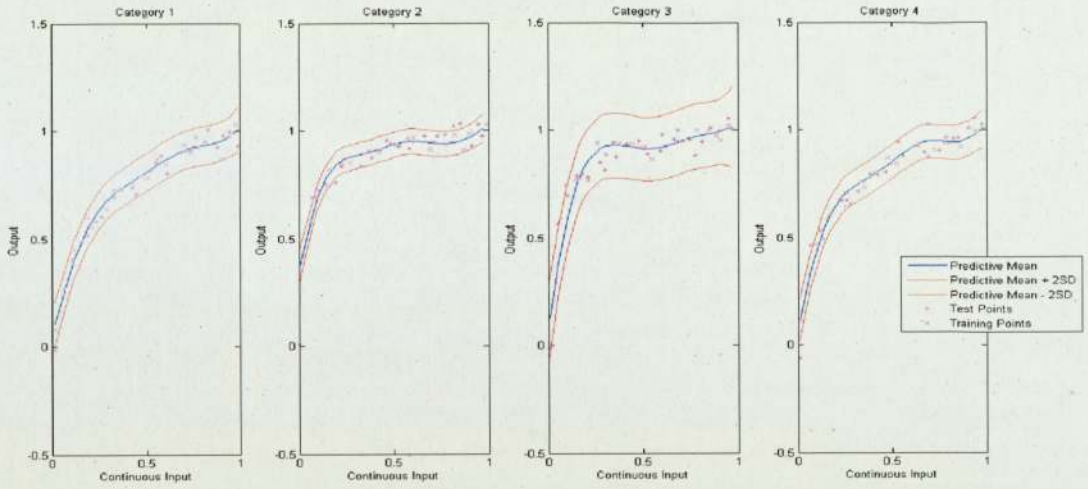


(b) Embedded GP Predictive Plot for Data set 7.

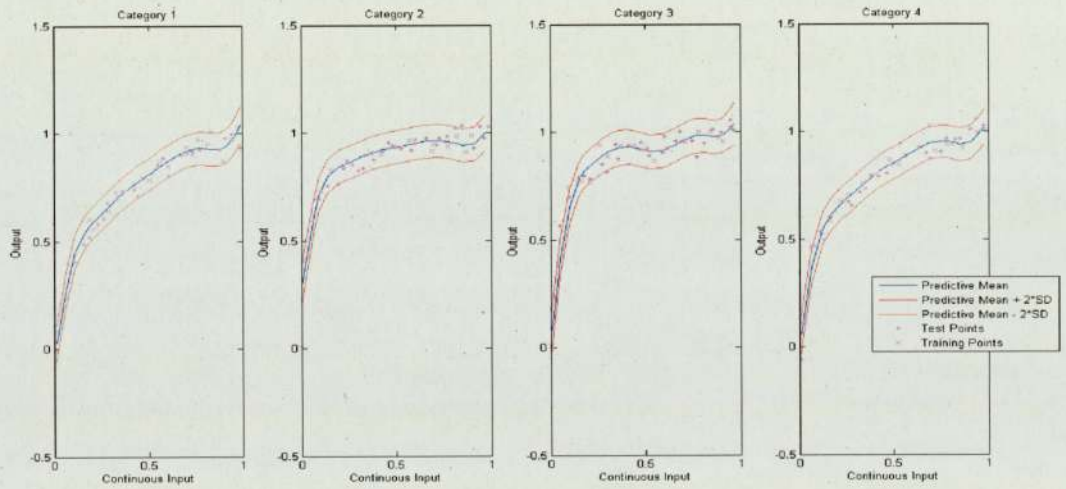


(c) Hypersphere GP Predictive Plot for Data set 7.

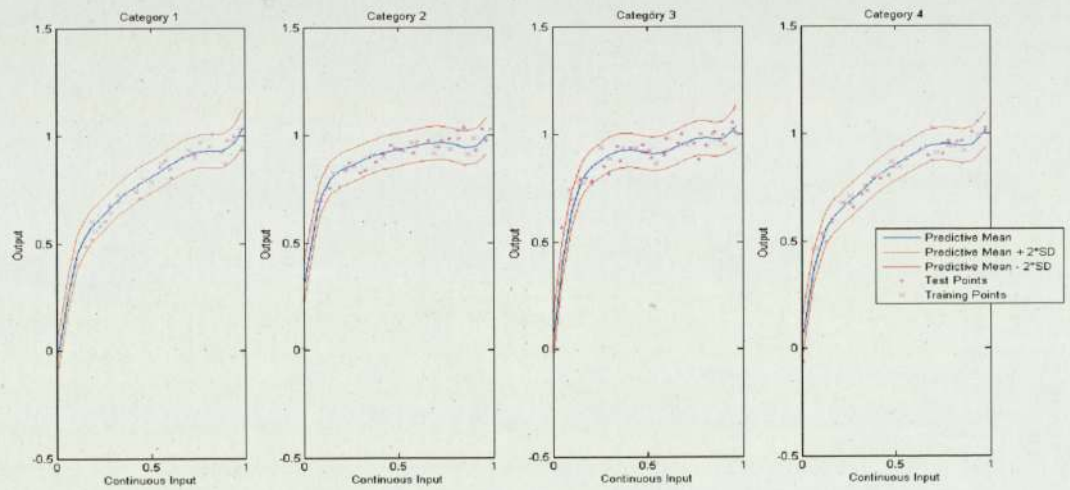
Figure 5.17: Predictive Plots for data set 7 using various GP models.



(a) Independent GP Predictive Plot per category for data set 13.

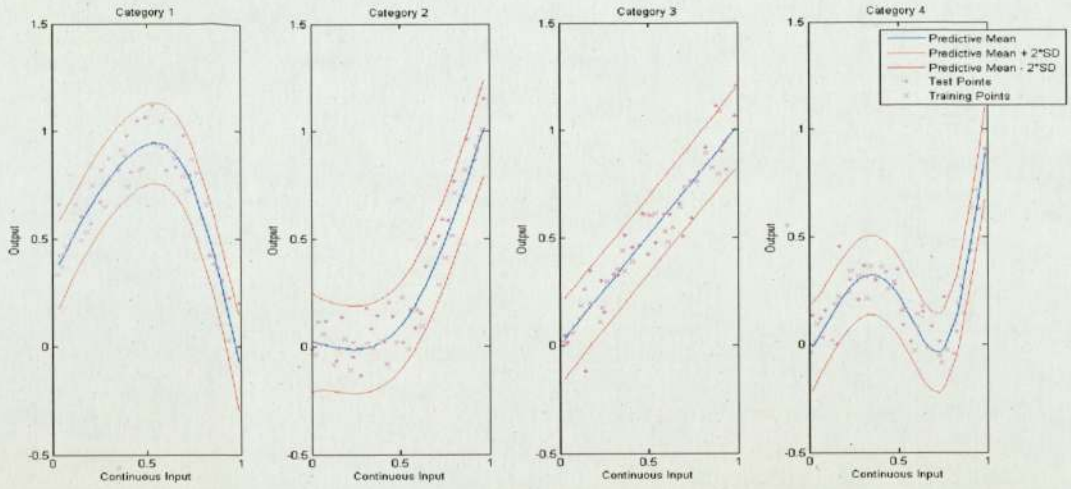


(b) Embedded GP Predictive Plot for data set 13.

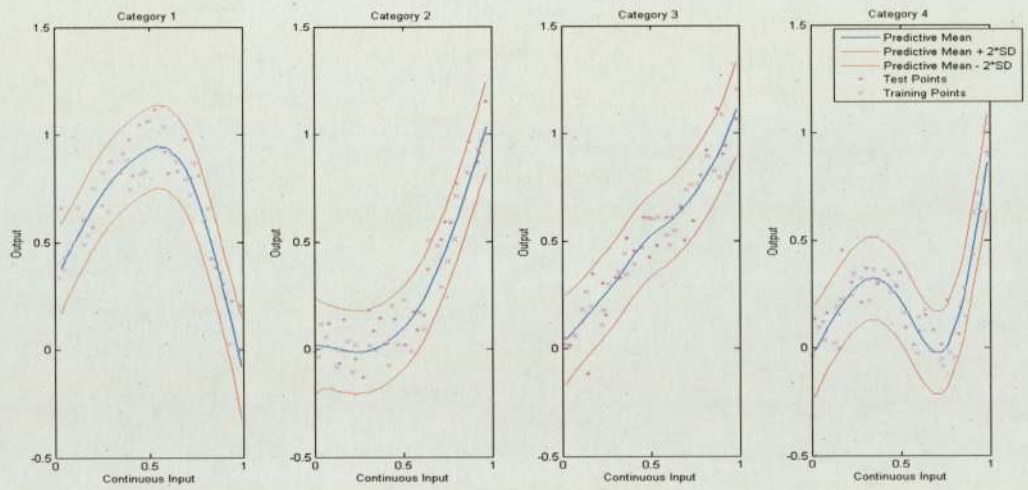


(c) Hypersphere GP Predictive Plot for data set 13.

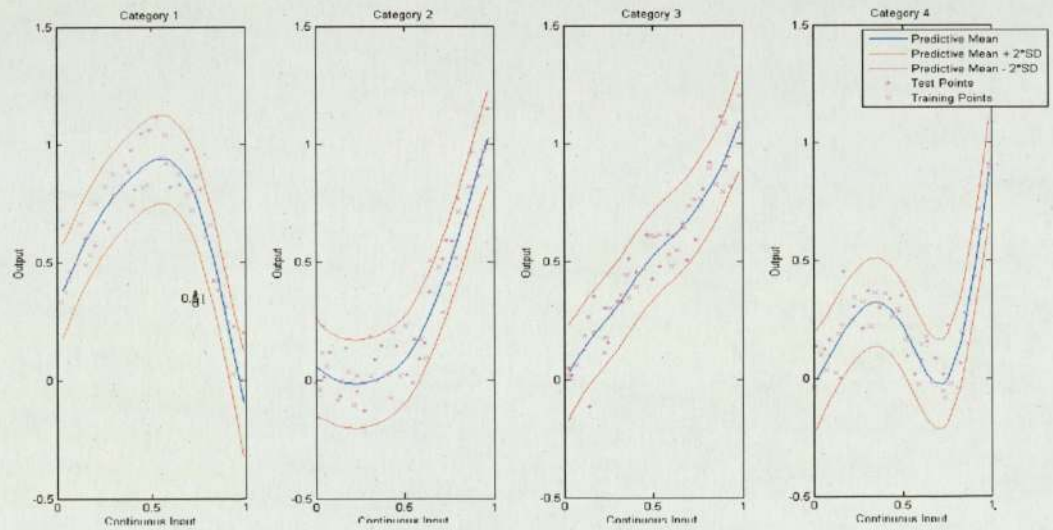
Figure 5.18: Predictive Plots for data set 13 using various GP models.



(a) Independent GP Predictive Plot per category for data set 1.

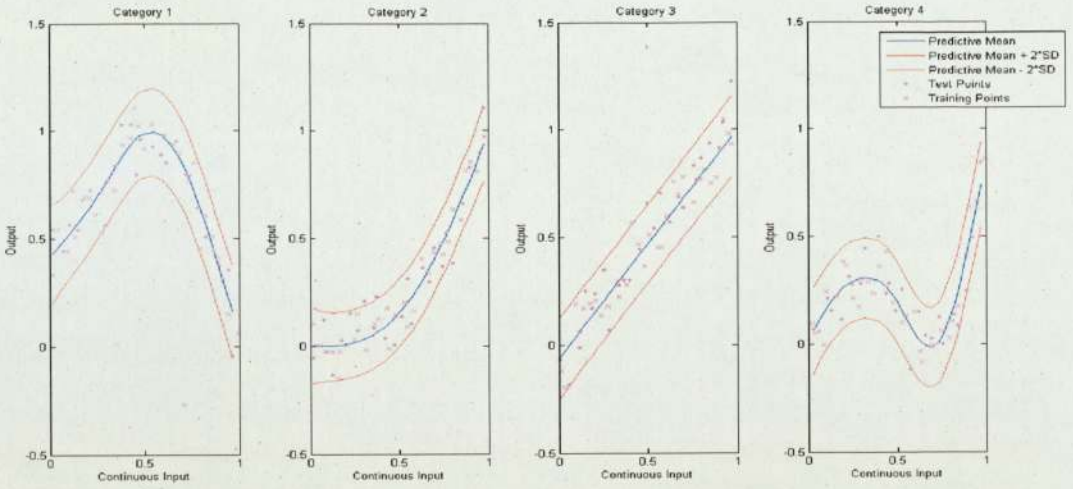


(b) Embedded GP Predictive plot for data set 1.

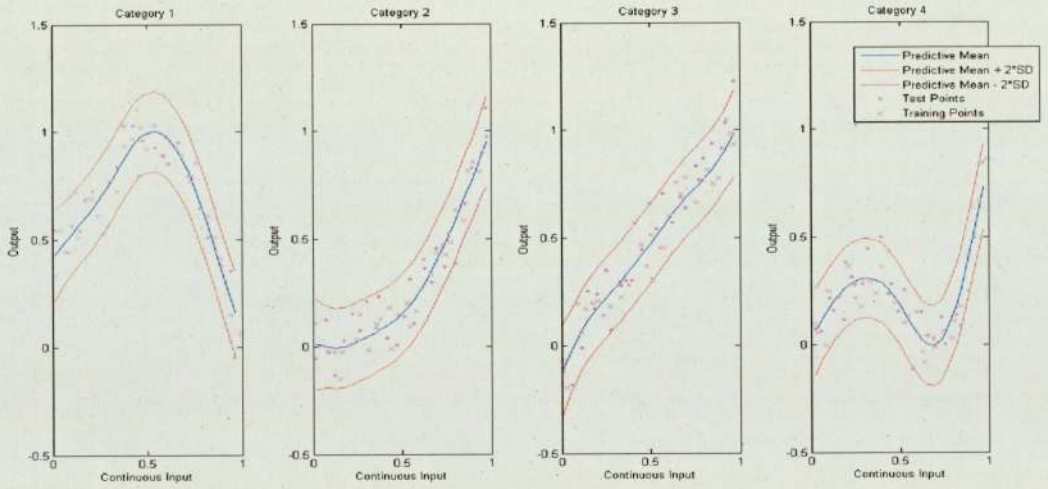


(c) Hypersphere GP Predictive Plot for data set 1.

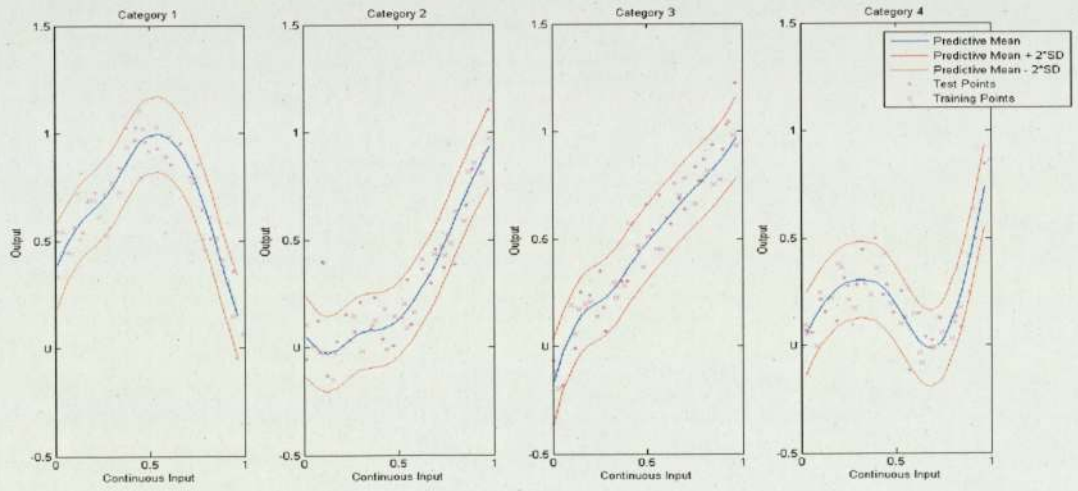
Figure 5.19: Predictive Plots for data set 1 using various GP models.



(a) Independent GP Predictive plot per category for Data set 2.

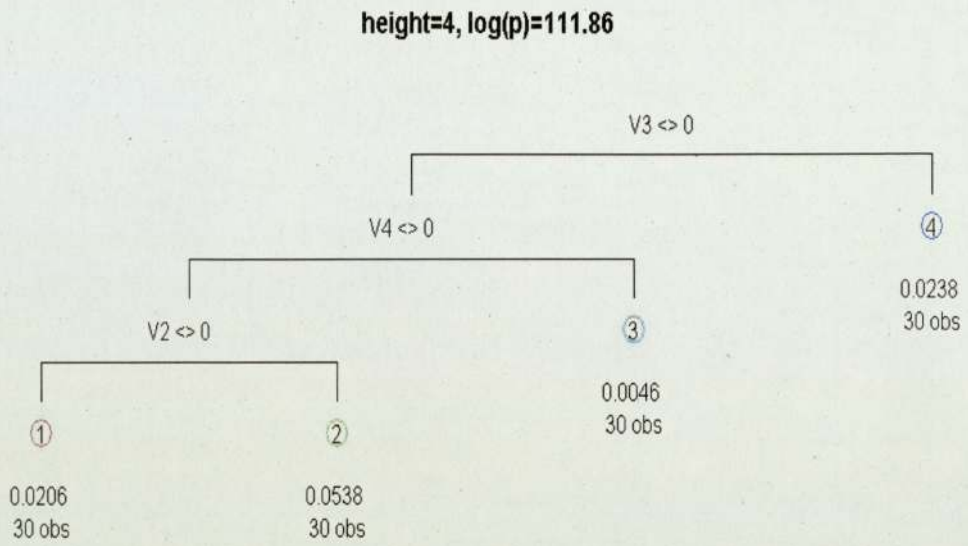


(b) Embedded GP Predictive plot for data set 2.

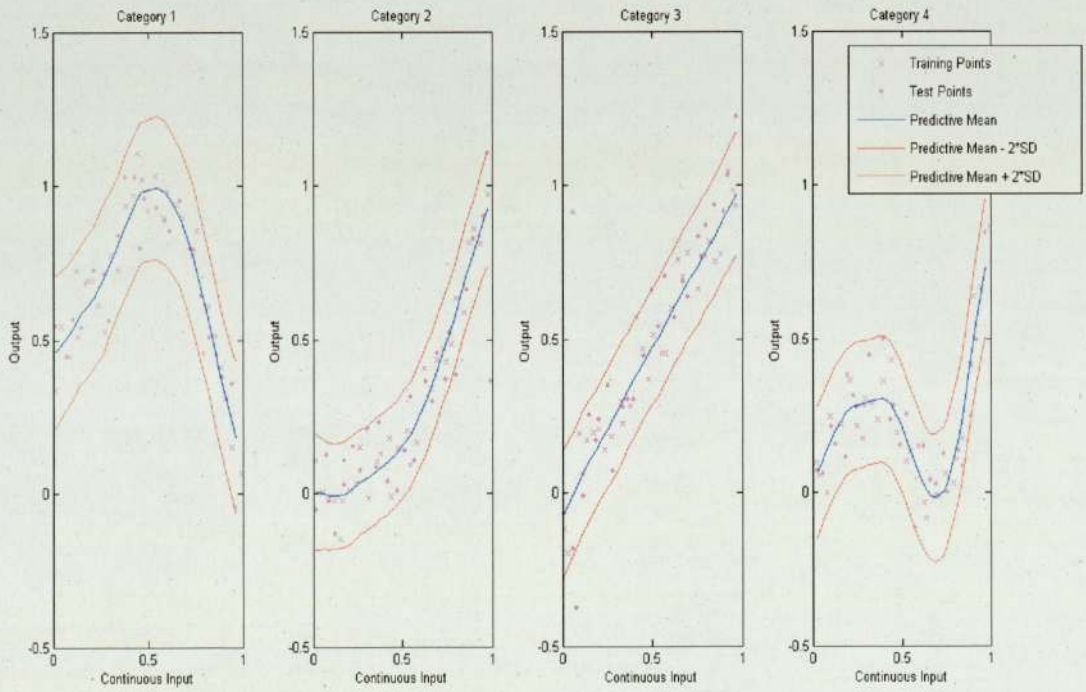


(c) Hypersphere GP Predictive plot for data set 2.

Figure 5.20: Predictive Plots for data 2 using various GP models.

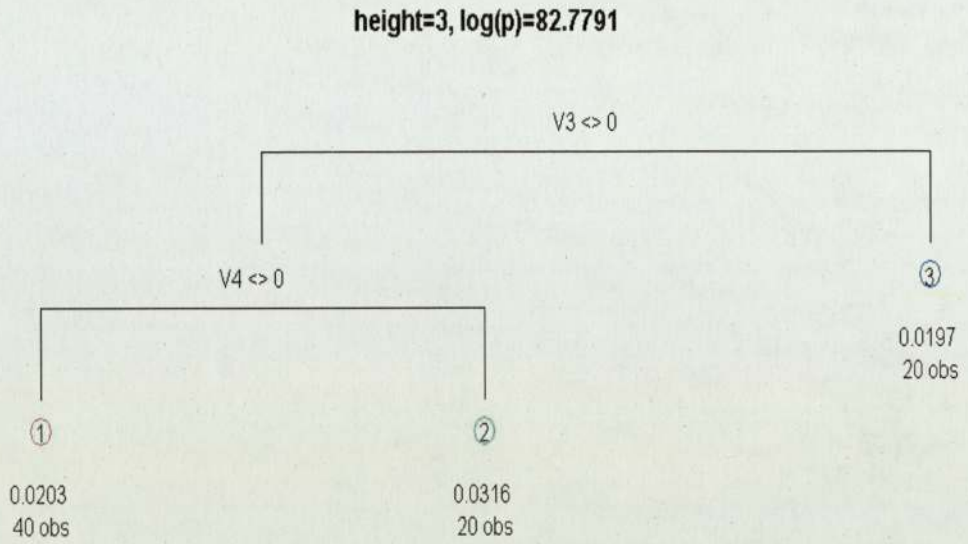


(a) Treed GP-LLM: Diagrammatic depiction of the maximum a *posterior* (MAP) tree using basemax= 1 and splitmin= 2 for data set 2;where 'obs'is the number of training points per a particular leaf; number above 'obs' is the noise variance for the model fitted at that leaf; V3,V4,V1 show partitioning on categories 2,3 and 1 respectively.

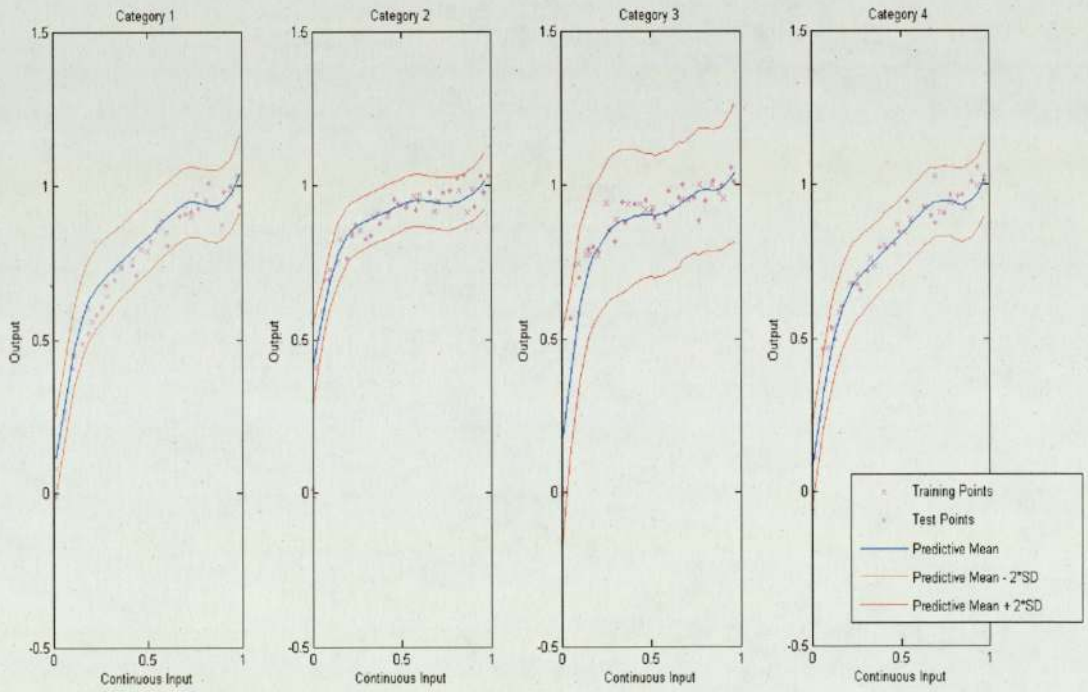


(b) Treed GP-LLM Predictive Plot for data set 2.

Figure 5.21: Treed GP-LLM Results for Data set 2.



(a) Treed GP-LLM: Diagrammatic depiction of the maximum a *posterior* (MAP) tree using basemax= 1 and splitmin= 2 for data set 13;where 'obs'is the number of training points per a particular leaf;number above 'obs' is the noise variance for the model fitted at that leaf; V3,V4 show partitioning on categories 2,3 respectively.



(b) Treed GP-LLM Predictive Plot for Data set 13.

Figure 5.22: Treed GP-LLM Results for Data set 13.

5.6 Experiment 6: Using Data Generated From The Flute Simulator

setting the problem

This experiment, involves comparing independent GPs (per category) against using the Embedded GP (ordinal) models, both which will use the data generated from the Flute simulator. Data sets were generated using multiple re-runs of the Flute simulator, where each run used a different value for R_0. The simulator was re-run 55 times, where R_0 could vary in the range 1.5 to 2.04 (in step sizes of 0.01). Hence each GP model, had access to 55 input-output pairs, $(\{\mathbf{x}_i, y_i\}, i = 1, ..., 55)$ for each category, hence the age group. This set is randomly split for six realizations, such that for each realization, twenty of those pairs were used for training whilst the other 35 were used for testing (per category). Outputs, f_i represent the cumulative amount of individuals infected (as a percentage of total individuals) for a particular age group on the final time step (day) of the simulation. Note that these responses are rescaled per category. Please refer to Section 5 for more details about the rescaling used.

output obtained using Flute Simulator

Figure 5.23 represents the cumulative number of infected individuals (as a percentage) on the final day of the simulation for each age group. This was obtained using multiple re-runs of the Flute simulator, where the (rescaled) output is what we would like to emulate using the various GP models, such as the Embedded (ordinal) model. Independent GPs (per category) and the Embedded GP (ordinal) models

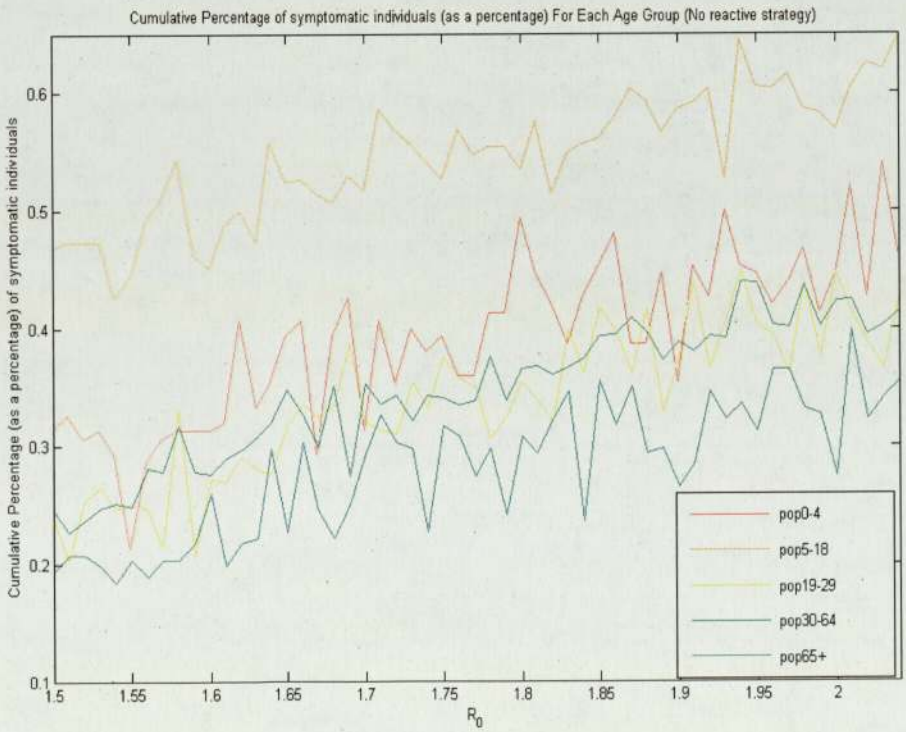


Figure 5.23: Cumulative number of infected individuals (as a percentage) for each age group.

parameters used in GP model

were used to emulate this output as shown in Figure 5.23. Hyper-parameters, $\boldsymbol{\theta}$ used in the Embedded GP (ordinal) framework were the length scale (ϕ), signal variance (σ_p^2), noise variance (σ_n^2) and each of the categorical mappings, $g_{1,k}, k = 1, 2, 3, 4, 5$ (from categories 1 to 5) to each age group. Each parameter in $\boldsymbol{\theta}$ is parameterized

as the form $\theta_b = e^{-m(b)}, b = 1, ..., 8$. Independent GP models however do not use the categorical mappings, and $\boldsymbol{\theta}$ only contained the length scale (ϕ), noise and signal variances (σ_n^2, σ_p^2 respectively) for this model. Separate mean functions were used per category in the GP prior, $p(\mathbf{y}|\boldsymbol{\theta}, X) = GP(m(.), k(.,.))$ for each GP model. Data points belonging to category i use the mean function of the following form, $\beta_{2i} + \beta_{2i-1}x_{i,1}$, where for $j = 1, ..., 2i-2$ and $j = 2i+1, ..., 2n_c$, ther terms β_j are multiplied by basis functions of 0. This is for the Embedded GP and Hypersphere GP frameworks. Independent GPs (per category) use the mean function of $\beta_1 + \beta_2 x_{i,1}$. The parameters, $\boldsymbol{\beta}$ is $2n_c \times 1$-dimensional for the Embedded GP and Hypersphere GP models, and 2×1-dimensional for independent GPs (per category) which have been analytically integrated out. Each β_i has a normal prior of N(0,100). Please refer to Appendix B and Section 2.5 for more details. Predictive equations, use $\hat{\boldsymbol{\theta}}$ which MAP. The Embedded GP (ordinal) model uses the following posterior distribution,

$$p(\boldsymbol{\theta}|X, \mathbf{y}) = \frac{(\int GP(\mathbf{y}|m, K) \prod_{i=1}^{2n_c} p(\beta_i|0, 100) \; d\boldsymbol{\beta})}{p(\mathbf{y}|X)} \times ...$$

$$... \times \frac{Ga(l|1.1, 1)Ga(\sigma_n^2|1.1, 1)Ga(\sigma_p^2|1.1, 1)(\prod_{i=1}^{3} Ga(g_{1,i}|1.1, 1))}{p(\mathbf{y}|X)}$$

Hyper-parameters, $\hat{\boldsymbol{\theta}}$ were found by MAP using SCG, where 100 different initializations for $\boldsymbol{\theta}$ were used to ensure the highest maximum of $p(\boldsymbol{\theta}|X, \mathbf{y})$ was found. The initial values for $\boldsymbol{\theta}$ were generated using $m_i \sim N(0, 1)$.

Each of the GP models, independent GPs (per category) and the Embedded GP use the Matern kernel (with order, $v = \frac{5}{2}$), where the co-variance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ used for each GP model are shown in Table 5.24. Validation scores ob-

Table 5.24: Co-variance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ used in each model (for Experiment 7).

-	Embedded GP (ordinal)	Independent GPs (per category)
$\text{Cov}(\mathbf{x}_i, \mathbf{x}_j)$	$\sigma_p^2 \left(1 + \sqrt{5}r + \frac{5r^2}{3}\right) \exp^{-\sqrt{5}r} + \sigma_n^2 I(\mathbf{x}_i, \mathbf{x}_j)$	$\sigma_p^2 \left(1 + \frac{\sqrt{5}r}{\phi} + \frac{5r^2}{3\phi^2}\right) \exp^{-\frac{\sqrt{5}r}{\phi}} + \sigma_n^2 I(\mathbf{x}_i, \mathbf{x}_j)$
r^2	$\frac{1}{\phi_1^2}(x_{i,1} - x_{j,1})^2 + (g_{1,k_1} - g_{1,k_2})^2$	$(x_{i,1} - x_{j,1})^2$

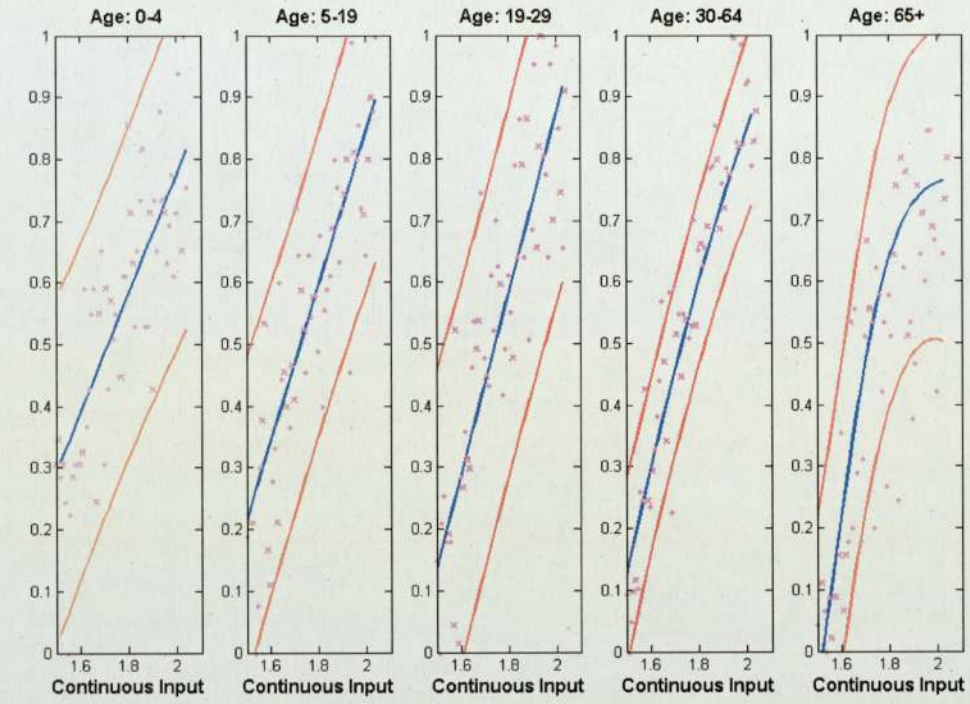
tained from using both GP models are shown in Table 5.25, where the best scores are in bold. Figure 5.6 shows the predictive plots obtained when using independent

Table 5.25: Validation measures obtained using the Embedded GP (ordinal) and Independent GPs (per category) frameworks for each training and test data set (Using Flute Simulator).

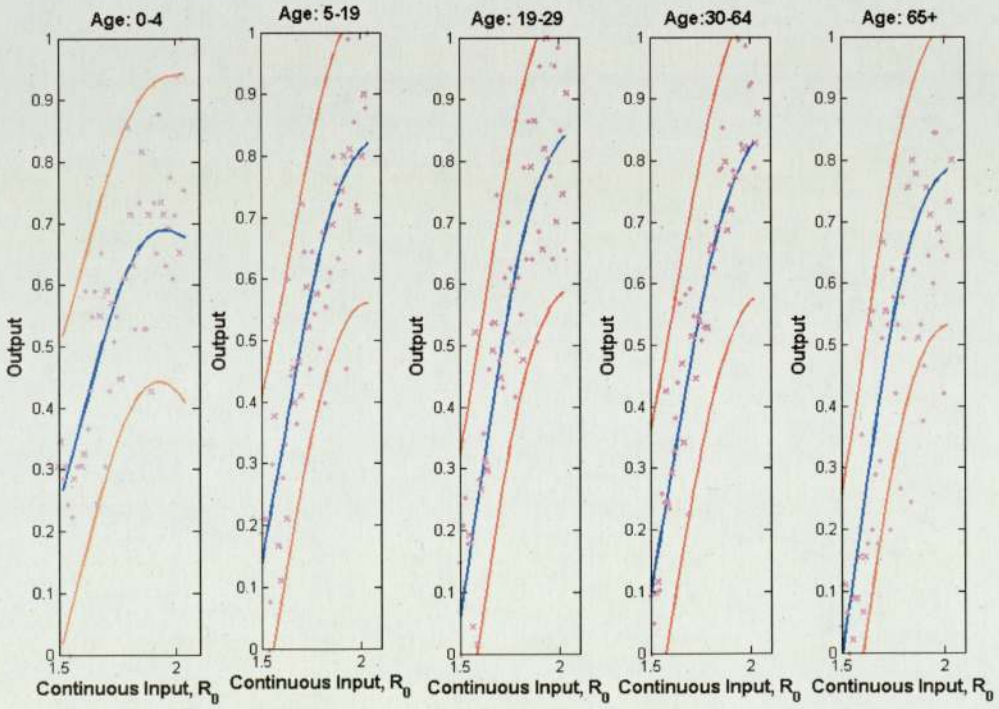
Embedded GP (ordinal)				Independent GP (per cat).			
Realization	MSE	NLPD	Dawid score	Realization	MSE	NLPD	Dawid score
1	**0.0178**	**-0.59**	**542.6**	1	0.0181	-0.53	527.8
2	**0.0137**	**-0.69**	**567.2**	2	0.0148	-0.65	554.6
3	**0.0182**	**-0.55**	**527.4**	3	0.0195	-0.51	513.0
4	**0.0153**	**-0.67**	**557.3**	4	0.0164	-0.61	540.1
5	**0.0151**	**-0.66**	**562.0**	5	0.0167	-0.64	553.7
6	**0.0166**	**-0.63**	**553.9**	6	0.0182	-0.59	545.3

GPs (per category) and the Embedded GP (ordinal) models for Realization 1. It is clear from examining the validation scores that the Embedded GP does better for all realizations than the independent GPs (per category). We believe this is because the responses from each category are similar. Multi-modality of the posterior, $p(\boldsymbol{\theta}|X, \mathbf{y})$ was not found on all realizations; this depends which data points are used for training. Another observation made is that independent GPs (for all these realizations) conclusions

per category decide to fit a linear line through the (training) data (for most of the categories), whilst the Embedded GP (ordinal) fits quadratic curves. This observation remains valid across all six realizations.



(a) Predictive distribution obtained using independent GPs (per category) for Realization 1.



(b) Predictive distribution obtained using Embedded GP (ordinal) for Realization 1.

5.7 Summary

summary

Results indicate that when responses are similar in each category, the Embedded GP (for either ordinal and nominal categorical variables), benefits from learning the embedded mappings between categories and reals, and yields better validation scores compared to using independent GPs per category. Using the Embedded GP, where the responses in each category are almost functionally independent, does not gain any benefit but rather yields validation scores similar to that of using independent GPs (per category). Validation scores in this situation are similar across all GP models. Treed GP-LLMs are not able to partition correctly, when responses are similar, and this could lead to other GP models, such as Embedded GP doing better. The Embedded and Hypersphere GP models have similar validation scores for most of the experiments, however this changes when the number of training data points in each category are reduced, (with a smaller noise variance). In this case, the Embedded GP model does significantly better than the Hypersphere GP. The additional benefit of using the Embedded GP model, is that the model uses less parameters than Hypersphere GP.

Embedded GP model can be applied to other tasks

The Embedded GP model does not necessarily need to be used for the regression task, but it can also be applied to other applications which will be reviewed in the next chapter, such as classification.

Chapter 6

Extensions

This chapter discusses how GPs (with no categorical mappings) can be used for binary classification before applying this framework to the Embedded GP model. Then, model selection will be reviewed for the purpose of reducing the number of categorical mappings used in the Embedded GP (nominal) model in the regression setting.

Goals of this chapter

6.1 Binary Classification

6.1.1 Binary Classification for Continuous Inputs

Throughout thesis so far, we have considered using GP models in the regression setting, but now we shall look at using these models for the classification task. We shall review how binary classification is applied to GP models which contain data points, which are continuous.

GPs used for binary classification involve placing a GP prior on the latent function, $f(\mathbf{x}_i)$, and squashing this through a logistic function, $\sigma(f_i)$, which is of the following form,

$$\sigma(f_i) = \frac{1}{1 + \exp^{-(f_i)}} \tag{6.1}$$

The probability that data input $\mathbf{x}_i$ has the output, $y_i = 1$ being equal to $\sigma(f_i)$, or probability 1-$\sigma(f_i)$ if data input $\mathbf{x}_i$ has output $y_i = 0$. Equation 6.1 is a possible form for the likelihood, $p(\mathbf{y}|\mathbf{f}, X)$. However, other likelihoods could also be used for classification, for example the probit function (with or without bias) and the step function (Rasmussen 2006, p.42). Throughout this section, the assumed likelihood is logistic.

different forms for the likelihood $p(\mathbf{y}|\mathbf{f}, X)$

GP models (used for classification) contain hyper-parameters, such as the length scale (ϕ), and signal and noise variances, (σ_p^2 and σ_n^2 respectively) which are contained in $\boldsymbol{\theta}$. The vectors $\mathbf{y}$, $\mathbf{f}$ are both $N \times 1$-dimensional vectors which contain the class labels and the latent values at each and every training data point respectively.

parameters used for GP models with continuous inputs

further notation

The role of $\mathbf{f}$ is to allow a convenient formulation of the model where the goal is to integrate out over the latent function, $\mathbf{f}$ (which is not observed), for prediction purposes. However, for classification it is not as straight forward as this, because the likelihood, $p(\mathbf{y}|\mathbf{f}, X)$ is no longer Gaussian. In the regression case, where the likelihood was a Gaussian distribution, $N(\mathbf{y}|\mathbf{f}, \sigma^2 I)$, various functions such as the posterior

role of f

comparisons between using GP models for regression and classification

distribution, $p(\boldsymbol{\theta}|\mathbf{y}, X)$ and predictive distributions could be calculated analytically. For the classification case, the likelihood is Bernoulli distributed, and has the form,

$$p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta}) = \prod_{i=1}^{N} p(y_i|f_i) = \prod_{i=1}^{N} \sigma(f_i)^{y_i}(1 - \sigma(f_i))^{1-y_i}.$$

The responses, y_i can take values either zero or one. As a result of this, the following integrals are used for prediction purposes,

$$p(f_i^*|X, \mathbf{y}, \mathbf{x}_i^*, \boldsymbol{\theta}) = \int p(f_i^*|X, \mathbf{x}_i^*, \mathbf{f}) p(\mathbf{f}|X, \mathbf{y}, \boldsymbol{\theta}) \, d\mathbf{f}, \tag{6.2}$$

$$p(y_i^*|X, \mathbf{y}, \mathbf{x}_i^*, \boldsymbol{\theta}) = \int \sigma(f_i^*) p(f_i^*|X, \mathbf{y}, \mathbf{x}_i^*, \boldsymbol{\theta}) \, df_i^*. \tag{6.3}$$

various ways of approximating the posterior density, $p(\mathbf{f}|X, \mathbf{y}, \boldsymbol{\theta})$

Both these integrals require either the posterior distribution, $p(\mathbf{f}|X, \mathbf{y}, \boldsymbol{\theta})$ to be approximated by a distribution, $q(\mathbf{f}|X, \mathbf{y}, \boldsymbol{\theta})$ or to obtain samples from it using MCMC. Two common forms for approximating $p(\boldsymbol{\theta}|X, \mathbf{y})$ are the Laplace approximation (Rasmussen 2006, p.41), and the Expectation Propagation (EP) method (Rasmussen 2006, p.52).

brief discussion about the EP algorithm

Using the EP method is more common in the literature because it gives good results (Rasmussen 2006, p.52), where instead of approximating the posterior probability by a single Gaussian, the likelihood is approximated using a local approximation scheme, where an un-normalized Gaussian function is assumed for f_i, such that,

$$p(y_i|f_i) \simeq Z_i N(f_i|\mu_i, \sigma_i^2),$$

where Z_i, f_i and σ_i^2 all indicate the site parameters, and the interest is finding out how the likelihood behaves as a function of f_i (Rasmussen 2006, p.53). The values for the site parameters can be optimized by minimizing the KL-divergence between the posterior and the approximate distribution (Rasmussen 2006, p.54). The Expectation Maximization-Expectation Propagation (EM-EP) is an extension of the EP-algorithm, where the E-step, involves finding the site parameters where the hyper-parameters, $\boldsymbol{\theta}$ are fixed, and the M-step involves maximizing the variational lower bound of $p(\mathbf{y}|\boldsymbol{\theta})$ given the approximate distribution for $p(\mathbf{f}|X, \mathbf{y}, \boldsymbol{\theta})$.

ways of optimizing or integrating out hyper-parameters $\boldsymbol{\theta}$

The EM-EP algorithm includes a way of optimizing both the hyper-parameters, $\boldsymbol{\theta}$ and the latent function values, $\mathbf{f}$. Other approaches in the literature, includes using a Laplace approximation to integrate out the latent function values, $\mathbf{f}$ and then using a Hybrid-Monte Carlo sampler to integrate out the hyper-parameters $\boldsymbol{\theta}$. Another approach involves using a Hybrid-Monte Carlo sampler to integrate out both the hyper-parameters and the latent function values. Further approaches are discussed in the paper by Kim & Ghahramani (2004). (Bishop 2006, p.315-318) suggests use Laplace approximation to integrate out the latent function values, and then optimize the hyper-parameters, $\boldsymbol{\theta}$, by maximizing the evidence.

We shall review the Laplace approximation in more detail for GP models involving continuous inputs only, as this approximation scheme will be used in a later example using the Embedded GP framework in the classification setting.

approximating the posterior by a Gaussian density

Laplace approximation is used because the likelihood, $p(\mathbf{y}|X,\mathbf{f})$, itself is non-Gaussian, and aids in the form of approximating the posterior, $p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$ by $q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$, a term which can be later used in Equations 6.2 and 6.3. The posterior distribution is approximated by a normal density,

$$p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta}) \simeq q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta}) = N(\mathbf{f}|\mathbf{w},A^{-1}),$$

where $\mathbf{w} = \arg\max_{\mathbf{f}} p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$ and $A = -\nabla\nabla \log p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})|_{\mathbf{f}=\mathbf{w}}$. The problem is finding the correct values for the terms, $\mathbf{w}, A^{-1}$. Using Bayes' formula, the posterior can be written as

$$p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta}) = \frac{p(\mathbf{y}|X,\mathbf{f})p(\mathbf{f}|X,\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{y}|X,\boldsymbol{\theta})},$$

specification of each component of the posterior density

where $p(\mathbf{y}|X,\mathbf{f})$ is the non-Gaussian likelihood, $p(\mathbf{f}|X,\boldsymbol{\theta})$ is a GP prior over the latent variables $\mathbf{f}$, and $p(\boldsymbol{\theta})$ is the prior over the hyper-parameters, $\boldsymbol{\theta}$. Because the prior, $p(\boldsymbol{\theta})$ and the normalization term of $p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$ do not depend on the latent variables, $\mathbf{f}$, this term will vanish when taking the first and second derivatives of $\log(p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta}))$ with respect to $\mathbf{f}$. Hence,

$$\nabla_{\mathbf{f}} \log(p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})) = \nabla \log(p(\mathbf{y}|\mathbf{f},X)) - K^{-1}\mathbf{f}, \tag{6.4}$$

$$\nabla\nabla_{\mathbf{f}} \log(p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})) = \nabla\nabla \log(p(\mathbf{y}|X,\mathbf{f})) - K^{-1} = -W - K^{-1}, \tag{6.5}$$

different form of likelihood implies different gradients of posterior density

where -$\nabla\nabla \log(p(\mathbf{y}|X,\mathbf{f}))$ is often denoted by W. The matrix, W, is diagonal because of the assumption that the non-Gaussian likelihood factorizes over each data point, $\mathbf{x}_i$. The derivatives for the log likelihood, $\log(p(\mathbf{y}|\mathbf{f},X,\boldsymbol{\theta}))$, depends on what likelihood function is used for classification. Before, proceeding to find the derivatives explicitly for $\nabla \log(p(\mathbf{y}|\mathbf{f},X))$, the following notation shall be introduced,

$$\boldsymbol{\sigma}_N = \begin{pmatrix} \frac{1}{1+\exp^{-f_1}} \\ \frac{1}{1+\exp^{-f_2}} \\ \cdots \\ \frac{1}{1+\exp^{-f_N}} \end{pmatrix},$$

where $\boldsymbol{\sigma}_N$ is a $N \times 1$ vector containing all evaluations of Equation 6.1 at each and every training point, $\mathbf{x}_i$. Likelihood probabilities are given by,

$$p(y_i = 1|f_i) = \frac{1}{1+\exp^{-f_i}},$$

$$p(y_i = 0|f_i) = 1 - \frac{1}{1+\exp^{-f_i}} = \frac{1}{1+\exp^{f_i}} = \frac{\exp^{-f_i}}{1+\exp^{-f_i}},$$

which then can be written in short-hand notation as

$$p(y_i = g|f_i) = \exp^{f_i g}\, \sigma(-f_i)\ ,\ g = \{0,1\}.$$

calculation of gradients of $p(\mathbf{y}|X,\mathbf{f})$

Using this knowledge, the first and second derivatives of the log likelihood term, $p(\mathbf{y}|X,\mathbf{f})$, which are given by,

$$\nabla_{\mathbf{f}} \log(p(\mathbf{y}|\mathbf{f},X)) = \mathbf{y} - \boldsymbol{\sigma}_N \tag{6.6}$$

$$\nabla\nabla_{\mathbf{f}} \log(p(\mathbf{y}|\mathbf{f},X)) = \begin{pmatrix} \sigma(f_1)(1-\sigma(f_1)) & 0 & \cdots & 0 \\ 0 & \sigma(f_2)(1-\sigma(f_2)) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \sigma(f_N)(1-\sigma(f_N)) \end{pmatrix}.$$

non-linearity equations to obtain $\mathbf{f}$

The mode of the log likelihood, is required for calculating the mean of the approximate distribution, $q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$. To find the optimal solutions for $\mathbf{f}$, one needs to combine Equations 6.4 and 6.6, and setting $\nabla_{\mathbf{f}}\log(p(\mathbf{y}|\mathbf{f},X))=0$. This however would yield a non-linear equation in latent variables, $\mathbf{f}$ because the terms, $\mathbf{f}$ appears also in vector $\boldsymbol{\sigma}_N$. In this case, a non-linear optimization routine such as Newton-Raphson (NR) should be used (Rasmussen 2006, p.43) to find the mode of the log posterior density, $p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$. NR used for this particular case, will use the iteration formula

Newton-Raphson to find optimal f

$$\mathbf{f}_N^{new}=\mathbf{f}_N^{old}-(\nabla\nabla\log(p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})))^{-1}\nabla\log(p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta}))$$

to find the optimal, $\mathbf{f}$.

obtaining the approximate posterior density

Once, NR has found the mode, $\mathbf{w}$, of the posterior distribution, $p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$, the covariance, A^{-1}, of $q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$ can be found by evaluating the Hessian of the negative log posterior at the mode, $\mathbf{w}$.

maximizing the variational lower bound of the log marginal likelihood

Once the approximation distribution, $q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$ has been found given the current values of $\boldsymbol{\theta}$, optimization over the $\boldsymbol{\theta}$ values is needed. We shall consider doing this by maximizing the variational lower bound whilst $q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$ is fixed, by using Jensen's inequality (Nickisch & Rasmussen 2008) as follows,

$$\log p(\mathbf{y}|X,\boldsymbol{\theta})=\log\int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X,\boldsymbol{\theta})p(\boldsymbol{\theta})\ d\mathbf{f}\geq\int q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})\log\frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X,\boldsymbol{\theta})p(\boldsymbol{\theta})}{q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})}\ d\mathbf{f},$$

where,

$$\int q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})\log\frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X,\boldsymbol{\theta})p(\boldsymbol{\theta})}{q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})}\ d\mathbf{f}:=P$$

or equivalently,

$$\log p(\mathbf{y}|X,\boldsymbol{\theta})\geq\int q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})\log p(\mathbf{y}|\mathbf{f})\ d\mathbf{f}+\int q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})\log(p(\mathbf{f}|X,\boldsymbol{\theta}))\ d\mathbf{f}\ldots+$$

$$\int q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})\log(p(\boldsymbol{\theta}))\ d\mathbf{f}-\int q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})\log q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})\ d\mathbf{f}\qquad(6.7)$$

The only terms which contributes in maximizing the variational lower bound is the second and third terms on the right-hand side of Equation 6.7. Calculating $\frac{\partial P}{\partial\theta}$ leads to

$$\frac{\partial P}{\partial\theta_i}=\frac{1}{2}\mathbf{w}'K^{-1}\frac{\partial K}{\partial\theta_i}K^{-1}\mathbf{w}-\frac{1}{2}tr(K^{-1}\frac{\partial K}{\partial\theta_i})-\frac{1}{2}tr(K^{-1}\frac{\partial K}{\partial\theta_i}K^{-1}(W+K^{-1})^{-1})+\frac{\partial\log p(\boldsymbol{\theta})}{\partial\theta_i}.$$

For optimizing over the hyper-parameters, $\boldsymbol{\theta}$, the scaled conjugate gradient algorithm can be used.

Iterative scheme

This is an iterative scheme, where one needs to optimize the latent values, $\mathbf{f}$ (given the current states for $\boldsymbol{\theta}$ which maximize the posterior density, $p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$) and then optimize over the hyper-parameters, $\boldsymbol{\theta}$ (maximizing the variational lower bound whilst the approximate distribution, $q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$ is fixed). This process is repeated until convergence was reached, for example, the absolute difference between optimal values of $\mathbf{f}$ between two iterative runs, j, $j+1$ was less than a certain tolerance level, ϵ. This is similar to an idea introduced by Chu & Ghahramani (2005).

evaluation of moments of predictive distribution, $p(f_i^*|X,\mathbf{y},\mathbf{x}_i^*,\boldsymbol{\theta})$

Once the optimum solutions for $\mathbf{f}$ and $\boldsymbol{\theta}$ have been found, the predictive mean and

variance of the distribution given in Equation 6.2 can be evaluated as follows,

$$E(f_i^*|X, \mathbf{y}, \mathbf{x}_i^*, \boldsymbol{\theta}) = k(\mathbf{x}_i^*, X)K^{-1}\mathbf{w},$$

$$Var(f_i^*|X, \mathbf{y}, \mathbf{x}_i^*, \boldsymbol{\theta}) = k(\mathbf{x}_i^*, \mathbf{x}_i^*) - k(\mathbf{x}_i^*, X)(K + W^{-1})^{-1}k(\mathbf{x}_i^*, X)^T.$$

Calculating the predictive probabilities for test points, $\mathbf{x}_i^*$

Because of the Gaussian approximation to the posterior distribution, $p(\mathbf{f}|X, \mathbf{y}, \boldsymbol{\theta})$, the predictive density $p(f_i^*|X, \mathbf{y}, \mathbf{x}_i^*, \boldsymbol{\theta})$, will be a Gaussian in this case. To calculate the probability of the class label at the test points being either 0 or 1, where this probability is indicated in Equation (6.3), samples can be generated from the Gaussian density, $N(f_i^*|k(\mathbf{x}_i^*, X)K^{-1}\mathbf{w}, k(\mathbf{x}_i^*, \mathbf{x}_i^*) - k(\mathbf{x}_i^*, X)(K + W^{-1})^{-1}k(\mathbf{x}_i^*, X)^T)$ (Rasmussen 2006, p.45), and using a crude Monte Carlo average,

$$p(y_*|X, \mathbf{y}, \mathbf{x}_*, \theta) = \frac{1}{i_c}\sum_{i=1}^{i_c}\sigma(s^{(i)})$$

highest probability assigned

where $s^{(1)}, s^{(2)}, ..., s^{(i_c)} \sim p(f_i^*|X, \mathbf{y}, \boldsymbol{\theta})$. A given test point, $\mathbf{x}_i$ is assigned to a class label, based on which probability of the class labels are the highest (Rasmussen 2006, p.45).

6.1.2 Embedded GP (Nominal) for classification

parameters used in the Embedded GP (nominal) for the classification task

The Embedded GP (nominal) model also uses the categorical one-to-one mapping values, $g_{1,j}, j = 1, ..., n_c$, where n_c is the number of categories (or n_c levels of qualitative factors). These parameters will be optimized along with the length scale (ϕ), and noise and signal variances, σ_n^2 and σ_p^2 respectively, where all these parameters are contained in vector, $\boldsymbol{\theta}$.

6.1.3 Example using the Embedded GP (nominal) model for classification

conditions used

We shall demonstrate using the Embedded GP (Nominal) model for binary classification, where the Matern kernel (with order, $v = \frac{5}{2}$) was used, where this took the same form as the one used in Experiment 4 from section 5, where there is one qualitative factor with 3 categories. All hyper-parameters were reparameterized as $\theta_i = e^{-m(i)}, i = 1, ..., 6$. The following posterior distribution was used

$$p(\mathbf{f}|X, \boldsymbol{\theta}, \mathbf{y}) \propto \left(\prod_{i=1}^{N}\exp^{f_i y_i}\sigma(-f_i)\right) GP(\mathbf{f}|\mathbf{0}, K)Ga(l|1.1, 1) \times ...$$

$$... \times Ga(\sigma_n^2|1.1, 1)Ga(\sigma_p^2|1.1, 1)\left(\prod_{i=1}^{3}N(g_{1,i}|0, 10)\right),$$

where y_i can take values 0 or 1. Data inputs have the following form,

$$\mathbf{x}_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ w_{i,1} \end{pmatrix},$$

where the first two components, $x_{i,1}, x_{i,2}$ are continuous whilst the other input, $(w_{i,1})$ is qualitative. There are an equal number of training and test points in each category, which are 15 and 25 respectively. The response (y_i) for each data point, is decided on the decision rule, $x_{i,1} \leq \frac{j}{4}$, where if the data point, $\mathbf{x}_i$ satisfies it the point belongs to class 0, or 1 otherwise. Variable j in this example stands for the category

which data point $\mathbf{x}_i$ belongs to. The Laplace approximation was used to find an approximation to the posterior distribution, and Jensen's inequality was used to maximize the variational lower bound, in order to find the optimal values for $\boldsymbol{\theta}$. The likelihood distribution, $p(\mathbf{y}|\mathbf{f}, X)$ was assumed to be a product of logistic functions, such that,

$$p(\mathbf{y}|\mathbf{f}, X) = \prod_{i=1}^{N} \sigma(f_i)^{y_i}(1 - \sigma(f_i))^{1-y_i}.$$

convergence and tolerance levels

The tolerance rate for Newton-Raphson's procedure used to find the optimum latent values $\mathbf{w}$, which maximizes $p(\mathbf{f}|X, \mathbf{y}, \boldsymbol{\theta})$ was set to 0.05 between two successive runs of the algorithm, whilst the SCG was set to a tolerance level of 10^{-5} for both the negative log posterior evaluation and the values for the hyper-parameters between two steps. Both these optimization routines, NR and SCG are applied iteratively, and the algorithm exits out of optimization if the optimum latent values, $\mathbf{w}$ between two trials of optimizing the hyper-parameters and the latent values satisfy

$$|\mathbf{w}_i - \mathbf{w}_{i+1}| \leq 0.05.$$

Figure 6.1 shows how the Embedded GP (nominal) model performed in this case. The misclassification rate for test points was given by $9.33\% = \frac{7}{75}$. The purpose

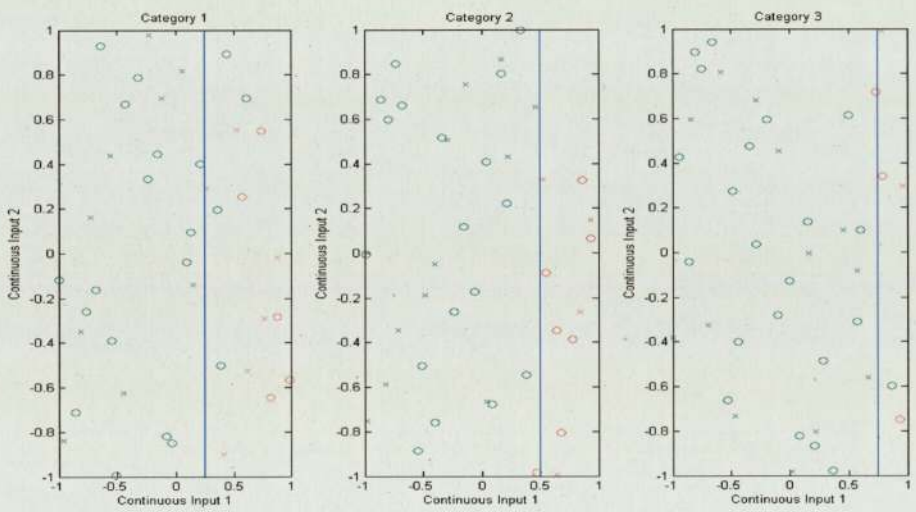


Figure 6.1: Graph showing the class label locations for both the training points (indicated by crosses), and prediction for test points (indicated by circles). Class labels 0 and 1 are indicated by colors green and red respectively, where the decision boundary is a straight line, $x_{i,1} \leq \frac{j}{4}$ (dependent on the category; for Categories 1-3).

results
main purpose

of this example was to illustrate that the Embedded GP (nominal) model can be adapted to classification tasks as well as the more focused regression task (seen in Chapters 1-5).

possibility of reducing the number of categorical mappings

The Embedded GP algorithm used for regression or classification tasks will use n_c variables, $g_{1,j}, j = 1, ..., n_c$. In the next section, we shall investigate, how the number of categorical mappings used in the Embedded GP could be reduced. This could be achieved by choosing a many-to-one mapping relationship between categories and real numbers.

6.2 Bayesian Model Selection Used For Embedded GP (nominal) models

possibility of having less parameters in the Embedded GP framework

Consider one qualitative factor, with n_c categories, where the responses for two categories are similar (or approximately the same). In this case, it would be sensible to place data from these two particular categories into the same group, because it would decrease the dimensionality of the parameter-space, which would will yield faster inference and calculations. But there are some considerations,

- Which of the n_c categories is placed in one group?
- Can there be more than one of these groups?
- Can the sizes of these groups be different?
- How does placing categories into groups affect the validation measures?
- How can models with different combinations (of groups and categories) be ranked?

infer a many-to-one relationship between categories and reals

Using each possible selection of groups in the Embedded GP (ordinal) model yields a particular model which can be used for testing (calculating the predictive equations and validating the model). Hyper-parameters used in the Embedded GP framework include ϕ (length scale for isotropic kernels), $\phi_i, i = 1, ..., n_d$ (length scales of separable kernels), noise (σ_n^2) and signal (σ_p^2) respectively, along with parameters, $g_{1,j}, j = 1, ..., C_{min}$, where $C_{min} \leq n_c$. Allocating data from n_c categories into C_{min} separate groups, and then learning the one-to-one mapping between groups and reals, infers a many-to-one relationship between the categories and reals.

ranking various combinations

We shall use the BIC (Bayesian Information Criterion) in order to rank model performance,

$$BIC_{M_i} = -2 \log p(\mathbf{y}|X, \boldsymbol{\theta}) + v \, ln(N),$$

comparisons between models performance

where N is the number of training points used and v is the dimensionality of hyper-parameter vector, $\boldsymbol{\theta}$ (Raftery 1993). There are other schemes such as the AIC (Akaike information criterion) in order to rank model performance. The BIC will be calculated for each (available) model, and judged by comparing the BIC scores between the different models. To compare models M_k and M_i, where M_k is larger than M_i consider the BIC measure, $E = BIC_{M_k} - BIC_{M_i}$, where the following criterion is used,

- E < 0 - Model M_k is favored; Reject model M_i,
- $0 \leq$ E ≤ 6 - Both models are favored; keeping both models, M_i and M_k,
- E > 6 - Model M_i is favored; Reject model M_k.

This criterion has been suggested by Raftery (1993), where this is analogy to the popular 5 percent significance level for tests.

analogy to the top-down algorithm

In order to pick the best model (which gives the lowest BIC), the following algorithm (where this is an analogy to top-down algorithm) will be used,

1. Calculate the BIC for full model (using all n_c categories; where there is single group per category)

2. Reduce the number of groups to $n_c - 1$. Consider all combinations; and evaluate each models BIC.
3. Iteratively compare the BICs across all these models removing the models which do not satisfy the criterion, as suggested by Raftery (1993). If two models are kept, keep one of the models aside and continue the algorithm with the chosen model.
4. Keep reducing the number of groups until one group is left.
5. At this point, there could be r remaining models.
6. Compare each of the remaining models; picking the models with the lowest BIC values (Posada & Buckley 2004).

We shall now apply Bayesian Model Selection to the Embedded GP (nominal) framework (used in the regression setting) by considering data generated from the Tree simulator. The first two categories, Douglas Fir (1) and Big Leaf Maple (2) for qualitative variable, *species*, will be considered first for testing purposes, before using the full set of possible categories.

6.2.1 Results: Applying Bayesian Model Selection

This methodology is applied to the data set generated in Experiment 1, where the first two categories, Douglas Fir (1) and Big Leaf Maple (2) are considered. The Matern kernel (with order $v = \frac{5}{2}$), where $k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_p^2 \left(1 + \frac{\sqrt{5}r}{\phi} + \frac{5r^2}{3\phi^2}\right) \exp^{-\frac{\sqrt{5}r}{\phi}} + \sigma_n^2 I(\mathbf{x}_i, \mathbf{x}_j)$ is used and the distance is defined as $r = (x_{i,1} - x_{j,1})^2 + g^2_{1,k_1} + g^2_{1,k_2}$. Note $g(.)$ represents the groups mapping value into the reals. The optimal values for the hyper-parameters, are chosen such that they maximize the log likelihood, $p(\mathbf{y}|X, \boldsymbol{\theta}) = GP(\mathbf{y}|0, K)$. conditions

There could be separate groups for each category, or both categories could be placed in the same group. The predictive plots for both options are shown below in Figures, 6.2 and 6.3. The modified top-down algorithm reports that having separate groups (for each of the categories) is better (as an indication of the BIC scores). Validation measures for each model is shown below in Table 6.1. involving only the Douglas Fir and Big Leaf Maple categories

Now, consider the same data set as before, but now with all four categories

Table 6.1: Two Categories; Bayesian Model Selection (Validation results).

Partition	BIC	-log($p(\mathbf{y}\|X, \boldsymbol{\theta})$	SMSE	NLPD	Dawid score
1,2	-99.4972	-58.9708	0.0307	-1.8013	275.1664
12,	-60.6380	-37.6967	0.0705	-1.3991	224.8627

are used. In this case, groups could be organized as, including all four categories

- 4 groups - 1 group per category
- 3 groups - 2 categories in one group; 2 categories in their own separate groups
- 2 groups - There are the following options:
  - Option 1. 2 categories in each of the groups
  - Option 2. 3 categories in one group, and a category in another

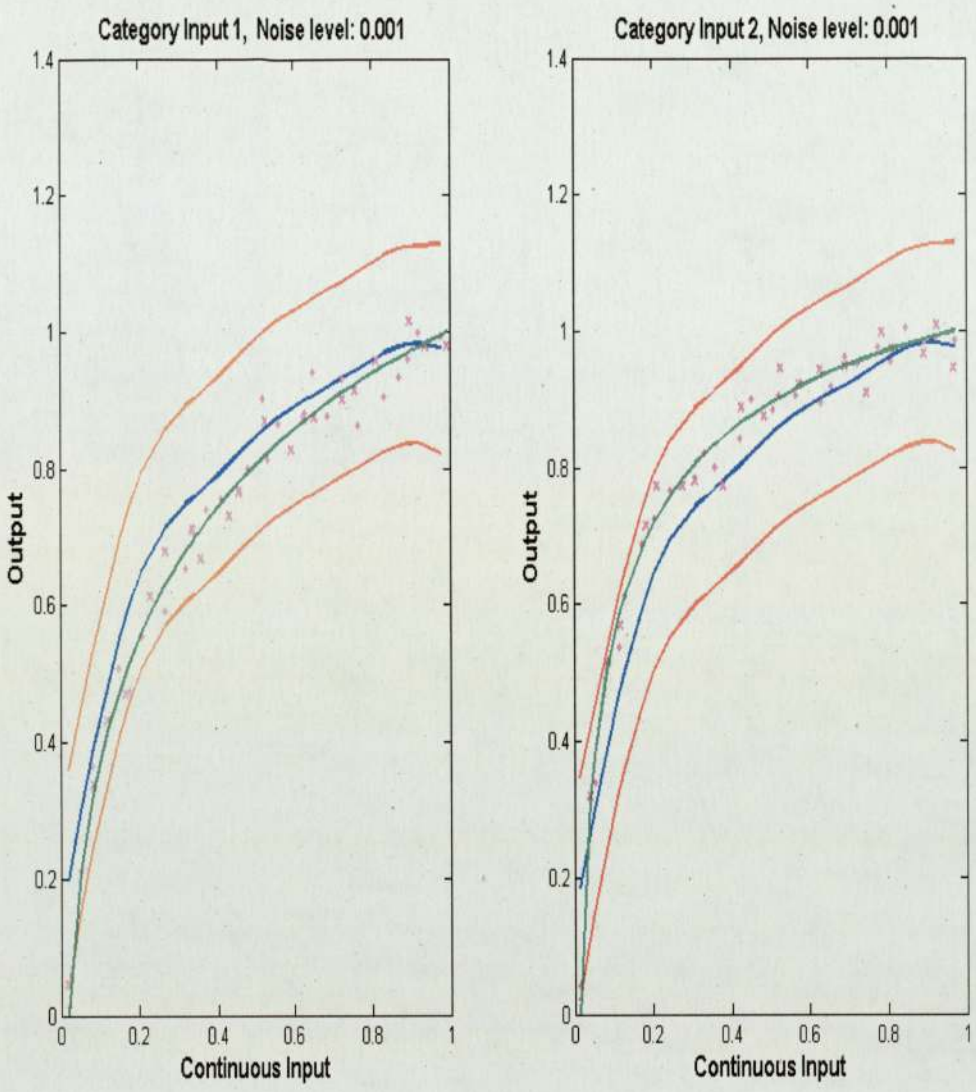


Figure 6.2: Best Model: Predictive plots, where two groups were used in the algorithm; one group per category.

- 1 group - All categories share the same group

Table 6.2 shows a summary of the validation measures of the most promising models. Predictive plots are shown in Figures 6.4 and 6.5 for the cases where there is a group per category and another case, where there are three groups. These are reported to be the best models according to the criterion above. Figures 6.3 and 6.5, show that by placing categories into the same groups, the resulting predictive measures will average the responses between categories.

Table 6.2: Four Categories; Bayesian Model Selection (Validation results).

Partition	BIC	$-\log p(\mathbf{y}\|X,\boldsymbol{\theta})$	SMSE	NLPD	Dawids Score
1,2,3,4	-193.4502	-112.0622	0.0274	-1.8463	559.5608
1,2,34	-196.5273	-111.4097	0.0349	-1.7345	534.7838

Averaging responses between categories

Model selection can be applied to the Embedded GP model, and can be seen as a

model selection reduces dimensionality of $\boldsymbol{\theta}$

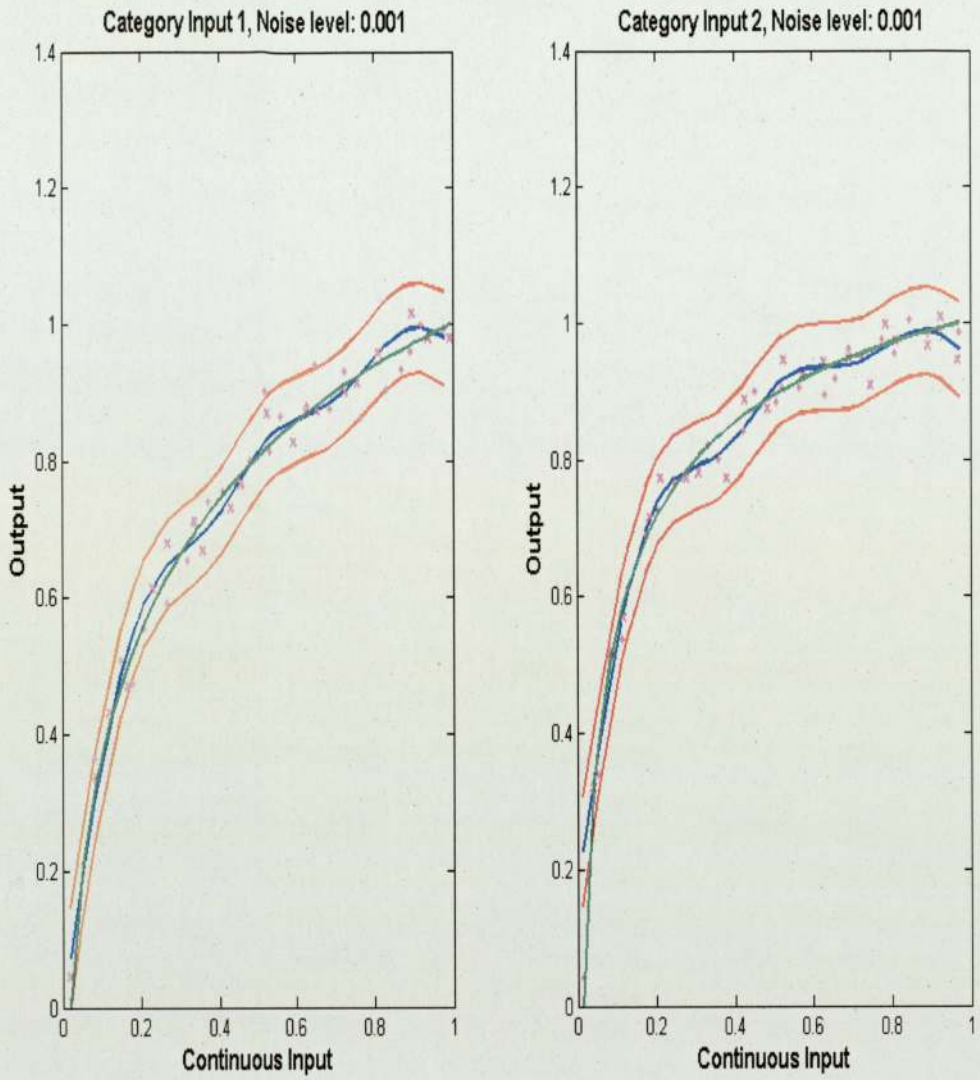


Figure 6.3: Predictive Plot: where one group was used in the GP model; both categories are placed in the same group.

suitable approach to reducing the number of parameters inferred in the Embedded GP framework. However, the modified top-down algorithm presented here could cause problems as the number of categories used in Embedded GP (nominal) framework increase. It may be computionally expensive to evaluate each and every model under each combination of chosen groups in this case.

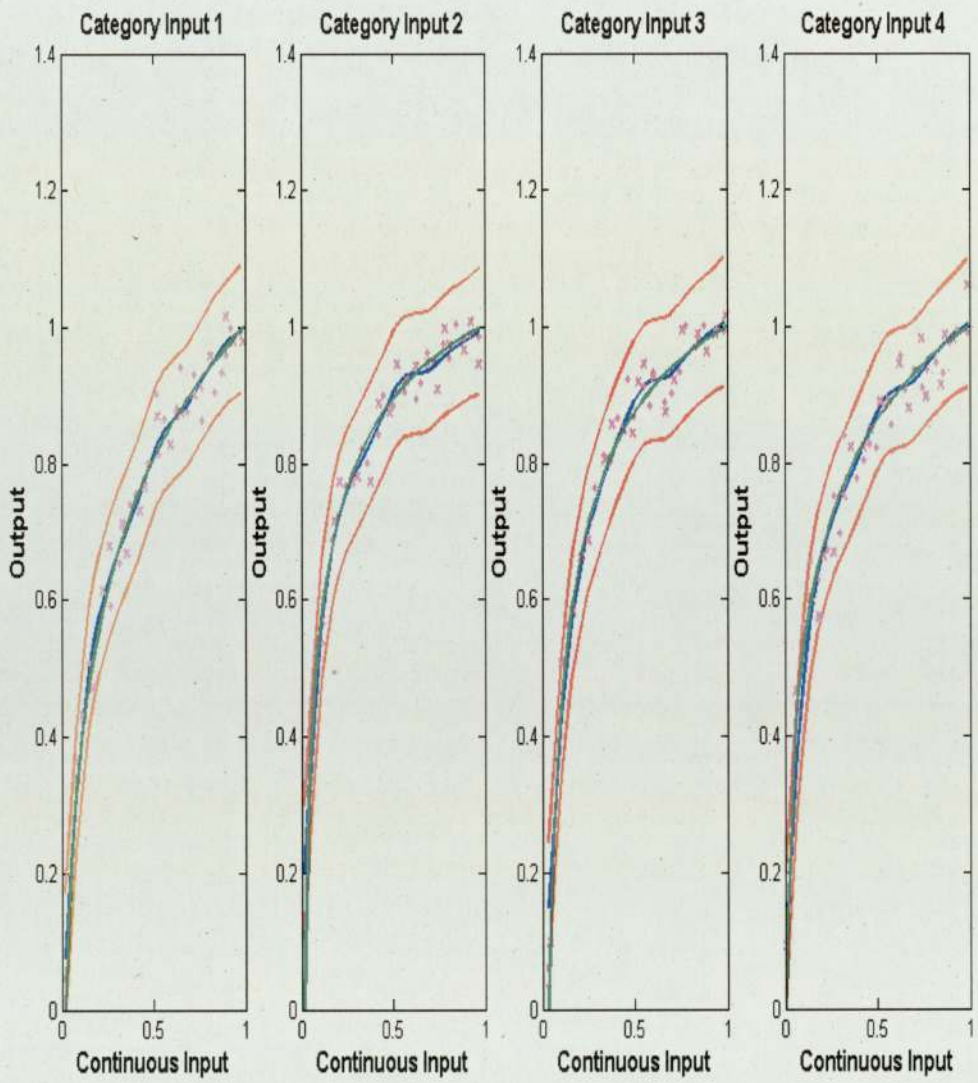


Figure 6.4: Best Model: Predictive Plot, where four groups were used in the Embedded GP model; one group per category.

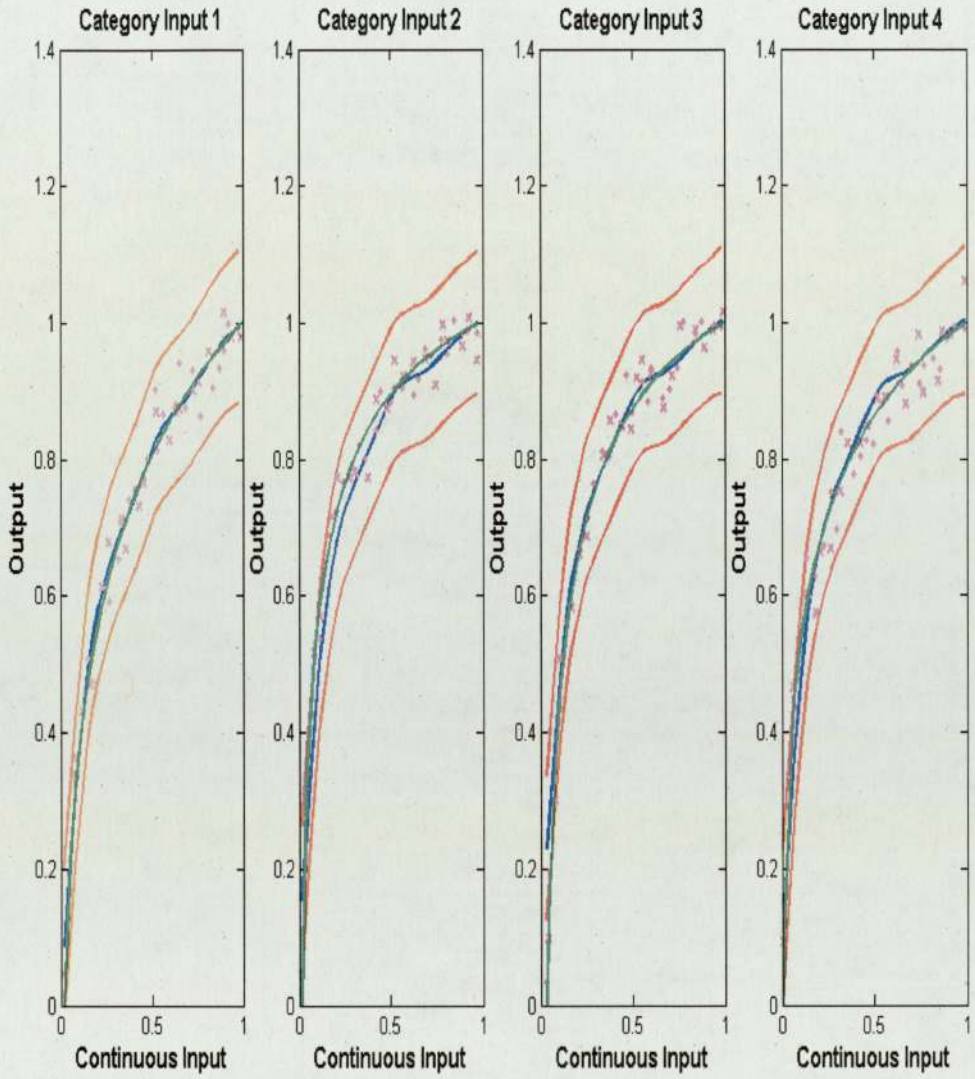


Figure 6.5: Best Model: Predictive Plot, where three groups were used in the Embedded GP model; where data belonging to categories 3, 4 (hence Willow and Bitter Cherry) were placed in one group whilst category 1 and 2 (hence Douglas Fir and Big Leaf Maple) placed in a group of their own respectively.

Chapter 7

Discussion, Key Findings and Future Work

7.1 Summary and Discussion

The aim of this project was to build a GP model (for the regression task), which was able to handle data inputs with continuous and categorical co-variates.

aim of the project/thesis

This thesis firstly reviewed the basic theory involving GPs used for regression where inputs only contain continuous entries in Chapter 2. This theory includes reviewing the posterior distribution, where various GP priors are used. Mean functions could also be incorporated in the GP prior using linear combinations of basis functions, $h_j(\mathbf{x}_i)$. Likelihood methods, such as MAP and ML could be used to find values for the hyper-parameters, $\boldsymbol{\theta}$, or a more Bayesian approach is to integrate out the uncertainty over these parameters. This chapter also reviewed sampling routines, such as Metropolis Hastings (MH) and Simulated Tempering (ST), which are used to evaluate integrals that are not analytically tractable. The two forms of MH, *block-wise* and *component-wise*, reviewed are sensitive to the step size of the transition steps. This could lead to being trapped in deep local minimum if these step sizes are too small. This is why ST is preferred over MH, because ST is able to escape local minima, by flattening the distribution which we are sampling from. Additionally, this chapter investigates various convergence diagnostics such as Batch-means and Potential Scale Reduction Factors (PSRF). By checking for these, ensures that the sampling routine has converged to the stationary distribution. This chapter, concludes with looking various validation scores, such as the MSE, NLPD, and Dawid score, which can be used to examine how good a GP model is at predicting responses at test data points, $\mathbf{x}_i^*$. These different scores use the predictive mean and variances, where these scores are used for accessing different things. The MSE uses only the predictive mean, to access the squared error between the predictive mean and the *actual* responses evaluated at each of the test points. Dawid score, however takes into consideration the full predictive covariance matrix, along with the predictive mean.

review of Chapter 2

Chapter 3 began with looking at the different types of categorical variables. The categorical variables reviewed, are *ordinal* and *nominal*. *Ordinal* categorical variables have an natural ordering between categories, whereas *nominal* categorical variables do not. This chapter then reviewed existing models in the literature, where qualitative co-variates are included as part of data inputs. These methods include Treed GP with Limiting Linear Models, and Hypersphere GP models. Treed GPs combine

review of Chapter 3

trees, which divide the predictor space based on splitting rules, and GPs which are flexible priors over functions, to form a powerful non-stationary approach to modelling. The Hyper-sphere GP, however models the correlation between categories by using a clever hypersphere parameterization. The Hypersphere GP model is able to capture negative and positive correlations between categories; an extension of the work done by Qian et al. (2008).

review of Chapter 4

Our approach to building a GP, (called the Embedded GP) considers different types of categorical variables, such as *ordinal* and *nominal*. The Embedded GP for ordinal categorical variables, uses a one-to-one mapping, placing categories onto a one-dimensional axis, whereas for ordinal categorical variables, categories are embedded using a 1-out-of-n_c encoding. This chapter, then compared the Embedded GP to other GP models reviewed in Chapter 3, in terms of advantages and disadvantages of using each GP model. Analytical comparisons between the Embedded GP and Hypersphere GP models, where the squared exponential kernel was used, indicated that the Embedded GP is restricted to having positive correlations between the responses of categories. However, the Embedded GP uses fewer parameters for categories than the Hypersphere GP model. Using independent GPs (per category) ignores the correlation between the responses of categories.

This chapter also gave some suggestions which should be considered when using the Embedded GP model in practice. These suggestions include using Cholesky decomposition for inversion of matrices such as kernel matrix, K, and using various parameterizations, such as e^{-m_i}, to ensure hyper-parameters remain positive. Specific implementations of the Embedded GP are included in Appendix B. These implementations were used for experimentation in Chapter 5.

review of Chapter 5

Chapter 5 compares various GP models such the Embedded GP, Hypersphere GP and Treed GP-LLMs along with Independent GPs (per category) on different data sets, where responses were generated using various simulators.

review of Chapter 6

Chapter 6 gave a brief introduction into how GPs are used for binary classification, where inputs, $\mathbf{x}_i$, only contain continuous entries. The likelihood, $p(\mathbf{y}|X,\mathbf{f})$ which is Bernoulli distributed. Approximation techniques for the posterior distribution, $p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$ were explored; and in specific the Laplace approximation. The distribution, $p(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$ (or its approximating distribution) was needed to calculate the predictive distribution at test points. Hyper-parameters, $\boldsymbol{\theta}$, appearing in the kernel, K of the GP prior were chosen based on maximizing the variational lower bound of the evidence (where distribution $q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$ is fixed). The EM-type algorithm used, was briefly reviewed, where this involves optimizing $q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$ and $\boldsymbol{\theta}$ until some convergence crietia has been reached. This framework was applied to the Embedded GP (nominal case) model for a simple example.

Other schemes were not discussed in detail, such as the EM-EP which could be used to find the approximate distribution, $q(\mathbf{f}|X,\mathbf{y},\boldsymbol{\theta})$ and hyper-parameters, $\boldsymbol{\theta}$.

Model selection was briefly discussed in this chapter, where this explored whether

categories could be embedded into the reals, using a many-to-one relationship. This was applied to a simple example using the Embedded GP (nominal) model, where different combinations of groups were ranked using the BIC criterion. For categories, having the same mapping into the reals, where the Embedded (nominal) GP uses prior $GP(\mathbf{f}|0, K)$, gvae the same predictive plots, where these indicate averaging the responses between these categories. Hyper-parameters in this case are optimized using MAP.

The aim of this chapter, was to be able to show that Embedded GP model can be extended to other tasks, such as classification and model selection.

7.2 Key Findings from Experiments

Key findings from Experiments

The following conclusions can be drawn from the experimentation performed in Chapter 5 when comparing the various GP models, Embedded GP, Hypersphere GP, Independent GPs per category and the Treed GP-LLM models for the regression task,

- Using data where the responses have been generated using the Tree simulator, has shown that using the Embedded GP is more appropriate than using separate GPs per category, which ignores correlations between the outputs of the categories. The Embedded GP has better validation scores, for the MSE, NLPD and Dawid score consistently for each data set used. This is because the responses from each category are similar and therefore it is beneficial to learn the correlation between the outputs of the categories. Treed GP-LLMs have difficulty in partitioning on the categories, giving different partition sizes for each data set tested on, where for some data sets this is appropriate whilst for others it could badly affect validation scores.

- Data sets, where the responses were generated using the Friedmann simulator have shown there is no benefit of learning the categorical mappings in the Embedded GP model. Both the Embedded GP and separate GPs per category gave very similar validation scores here in terms of MSE, NLPD and Dawid score. This is because the responses from each category are (almost) unrelated. Treed GP-LLMs were able to partition on the categories correctly, giving validation scores close to that the Embedded GP and Hypersphere GP models.

- By integrating out the hyper-parameters, $\boldsymbol{\theta}$ in Equation 2.16, does not significantly improve the quality of the validation scores, otherwise offered by MAP (shown in Experiment 2 in Chapter 5) for the generated data set.

- The Embedded GP and Hypersphere GP models perform very close in validation scores (for most) of the experiments that they are compared. Differences in these two GP models emerge when using twenty (training) data points per category (with a lower noise level), where the Embedded GP performs better of the two GP models. This could be due to the amount of parameters used in each of these models.

- When using any of the GP models, ensure multiple instances of $\boldsymbol{\theta}$ are used to find the maximum of the posterior, $p(\boldsymbol{\theta}|X, \mathbf{y})$ or likelihood, $p(\mathbf{y}|X, \boldsymbol{\theta})$, as there is a risk of multi-modality. Note, that these parameter estimates, $\hat{\boldsymbol{\theta}}$ are then used in the predictive distribution. GP models such as the Hypersphere GP, independent GPs (per category), Embedded GP (for ordinal and nominal categorical inputs) which use the posterior (or likelihood) surfaces, were often

multi-modal. Posterior surfaces are multi-modal, due to the training data, $(X, \mathbf{y})$. Multiple initializations ensure the highest possible maximum (and even possibly the global maximum) is found.

7.3 Future Work

Future Work

The future work that could extend the results presented in this report is outlined in the list below.

- Each of the GP models used in Chapter 5 should be applied to more real-life data set examples. This is to ensure that the following arguments made in the previous section are consistent.

- Use model selection to decrease the number of parameters used for inference of the Embedded GP (both ordinal and nominal setting), where this will be necessary when using the Embedded GP with a large number of categories. Exploring all possible models exhaustively will be computationally expensive, because the number of models to be explored grows exponentially as the number of categories increase. Similar responses from different categories could be explained by using less variables.

- Throughout this thesis, there as been an assumption that all GP models, such the Embedded GP, or Independent GPs per category are built independently. Another interesting extension, would be to create a sequential framework, where for D categories, data corresponding to D_1 categories are used for the Embedded GP model, where $D_1 < D$. For data corresponding to the remaining categories either, independent GPs (per category) could be built or another Embedded GP could be fitted using data from a subset of the remaining categories. However, other issues would need to considered here for instance,

  - How many Embedded GP models should we use?
  - How many categories should be used for each Embedded GP model?

- Test each of the GP models, the Embedded GP, Hypersphere GP, and Independent GPs (per category) using a different kernel functions. Throughout Chapter 5, the experiments use a Matern covariance kernel (with order $v = \frac{5}{2}$), because of its numerical stability for covariance matrix, K. Different kernel functions could possibly have different results.

- Test the Embedded GP (Nominal) model where there are different process variances σ_p^2 for each category which would add a further d parameters to be estimated through MAP or ML. No separate signal variance for the cross categories, because of the categorical mappings, $g_{1,j}, j = 1, ..., n_c$, would enter into the covariance, $k(\mathbf{x}_i, \mathbf{x}_j)$. Separate signal variances (per category), would give the Embedded GP more flexibility. The additional benefit of having separate signal variances per category, is that the pre-processing step of rescaling the responses (per category) could be removed. NB: This model had been built, but there was not enough time to test this model.

- Another possible extension, would be to use a different embedding for the Embedded GP framework. In this thesis, we have considered embedding categories onto a 1-D axis (for *ordinal* categories) and n_c-dimensional space (for n_c *nominal* categorical variables) using the 1-out-of-n_c encoding. Suppose now embedding *nominal* categories into a 2-D space, where each category is associated with a

two-dimensional vector, (s_1, s_2). This would have the advantage of not assigning categories directly onto the axis. Note, that the Embedded GP (nominal) model as it has been constructed using the 1-out-of-n_c encoding, cannot handle groups of categories, which have similar responses. If responses from different categories are similar, then using the Embedded GP (nominal) model, yields mapping values close to zero (hence forming a cluster of categories).

Appendices

Appendix A

Design Input Issue: Example

Suppose there is one qualitative factor, where there are two possible categories, 'Sean' and 'Mike' , which both have a categorical mapping value of 0. The kernel matrix, K, is given by,

$$K = \begin{pmatrix} k\left(\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix}\right) & k\left(\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}2\\0\end{pmatrix}\right) & k\left(\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix}\right) & k\left(\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}2\\0\end{pmatrix}\right) \\ k\left(\begin{pmatrix}2\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix}\right) & k\left(\begin{pmatrix}2\\0\end{pmatrix},\begin{pmatrix}2\\0\end{pmatrix}\right) & k\left(\begin{pmatrix}2\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix}\right) & k\left(\begin{pmatrix}2\\0\end{pmatrix},\begin{pmatrix}2\\0\end{pmatrix}\right) \\ k\left(\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix}\right) & k\left(\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}2\\0\end{pmatrix}\right) & k\left(\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix}\right) & k\left(\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}2\\0\end{pmatrix}\right) \\ k\left(\begin{pmatrix}2\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix}\right) & k\left(\begin{pmatrix}2\\0\end{pmatrix},\begin{pmatrix}2\\0\end{pmatrix}\right) & k\left(\begin{pmatrix}2\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix}\right) & k\left(\begin{pmatrix}2\\0\end{pmatrix},\begin{pmatrix}2\\0\end{pmatrix}\right) \end{pmatrix}.$$

determinant

principal minor $K_{3\times3}$

The determinant of matrix, K, which excludes the elements from the last row and column, has a determinant of zero. The matrix K has broken the rules of positive definiteness, where this requires that all principal minors have determinants that are strictly positive. The matrix, K needs to be positive definite, in order to perform cholesky decomposition on it in MATLAB. This problem would occur even if different initial mapping values for each of the categories were used (but the same for both categories), because the design input points have been chosen poorly in the algorithm.

Appendix B

Various Implementations of GP models

This section outlines the more specific implementations of the GP models used for experimentation in Chapter 5.

B.1 Embedded GP (ordinal case) for Regression with GP prior, $GP(0, K)$

Given training and test data, $(X, \mathbf{y})$ and $(X^*, \mathbf{y}^*)$ which has n_d continuous covariates and n_c categories (or n_c combinations of the qualitative factors). Note there are j_D qualitative factors. The following steps are taken for this algorithm,

steps for Embedded GP model (GP prior with no mean)

1. Gather all training and test data points which have the same qualitative categorical inputs. Choose kernel, k which will be used in Embedded GP model. The total number of training and test points to be used in the model altogether is given by Q_{tr} and Q_{te} respectively.

2. Given n_c different combinations of qualitative factors, assign each combination to a hyper-parameter $g_{1,j}\ \forall j$, where$1 \leq j \leq n_c$.

   This transforms data input,

   $$\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n_d}, w_{i,j_1}, w_{i,j_2}, ..., w_{i,j_D})' \rightarrow \mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n_d}, g_{1,j})'.$$

   The transformed input will be used in the Embedded GP.

3. Hyper-parameters, σ_n^2 (noise variance), σ_p^2 (process variance), $\phi_i, i = 1, ..., n_d$ (length scales for separable kernels), ϕ (length scale for isotropic kernels) and $g_{1,j}, j = 1, ..., n_c$ all re-parameterized to a form such as m^2, $\exp^m$ to ensure the parameters are constrained to be positive. All hyper-parameters are contained in $\boldsymbol{\theta}$.

4. Assign an initial value for each hyper-parameter in $\boldsymbol{\theta}$ using samples from Gaussian distributions.

5. Use an optimization routine such as, scaled conjugate (SCG) to minimize,

   $-\log p(\boldsymbol{\theta}|X, \mathbf{y}) = -1 \times ((-0.5 \times e) - (0.5 \times (\mathbf{y}' \times (A' \setminus (A \setminus (\mathbf{y}))))) - \frac{Q_{tr}}{2} \times \log(2 \times \pi)) - \log(p(\boldsymbol{\theta}))$,

in respect to $\boldsymbol{\theta}$, where $e = \sum_{i=1}^{Q_{tr}} \log \lambda_i$, $\lambda_i = eig(K(X,X))$, $A = chol(K(X,X))'$. A is an upper triangular matrix has a result of performing Cholesky decomposition on $K(X,X)$. Prior distribution over the hyper-parameters defined as, $p(\boldsymbol{\theta})$ assumes to be factorizable over each hyper-parameter, θ_j.

6. Once having obtained $\hat{\boldsymbol{\theta}}$ which maximizes distribution, $p(\boldsymbol{\theta}|X,\mathbf{y})$, use $\hat{\boldsymbol{\theta}}$ to transform the test points such that,

$$\mathbf{x}_i^* = (x_{i,1}^*, x_{i,2}^*, ..., x_{i,n_d}^*, w_{i,j_1}^*, w_{i,j_2}^*, ..., w_{i,j_D}^*)' \rightarrow \mathbf{x}_i^* = (x_{i,1}^*, x_{i,2}^*, ..., x_{i,n_d}^*, \hat{g_{1,j}^*})'.$$

7. Use remaining hyper-parameters, σ_n^2, σ_p^2, $\phi_i, i = 1, ..., n_d$ (for separable kernels), ϕ (for isotropic kernels) to evaluate moments of the predictive distribution, $p(y_i^*|\mathbf{y}, X, \mathbf{x}_i^*, \boldsymbol{\theta}_{opt})$, where again Cholesky decomposition is used,

   - $E(y_i^*|\mathbf{y}, X, \mathbf{x}_i^*, \hat{\boldsymbol{\theta}}) = cov(\mathbf{x}_i^*, X) \times (A' \setminus (A \setminus (\mathbf{y}')))$,
   - $Var(y_i^*|\mathbf{y}, X, \mathbf{x}_i^*, \hat{\boldsymbol{\theta}}) = cov(\mathbf{x}_i^*, \mathbf{x}_i^*) - cov(\mathbf{x}_i^*, X) \times (A' \setminus (A \setminus cov(X, \mathbf{x}_i^*))$.

suggestions

Note that,

- If the minimization of the likelihood, $p(\mathbf{y}|X,\boldsymbol{\theta})$ is required minimize, $-\log p(\mathbf{y}|X,\boldsymbol{\theta}) = -1 \times ((-0.5 \times e) - (0.5 \times (\mathbf{y}' \times (A' \setminus (A \setminus (\mathbf{y})))) - \frac{Q_{tr}}{2} \times \log(2 \times \pi))$ instead of using Step 5.
- in the case of nominal case, transform data input,

  $\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n_d}, w_{i,j_1}, w_{i,j_2}, ..., w_{i,j_D})' \rightarrow$
  $\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n_d}, g_{1,1}\delta_1, g_{1,2}\delta_2, ..., g_{1,j}\delta_j, ..., g_{1,n_c}\delta_{n_c})'$ in place of Step 2 and replace Step 6 by,
  $\mathbf{x}_i^* = (x_{i,1}^*, x_{i,2}^*, ..., x_{i,n_d}^*, w_{i,j_1}^*, w_{i,j_2}^*, ..., w_{i,j_D}^*)' \rightarrow$
  $\mathbf{x}_i^* = (x_{i,1}^*, x_{i,2}^*, ..., x_{i,n_d}^*, \hat{g_{1,1}}\delta_1, \hat{g_{1,2}}\delta_2, ..., \hat{g_{1,j}}\delta_j, ..., \hat{g_{1,n_c}}\delta_{n_c})'$.
  The notation, $\delta_j = 1$ if the data point belongs to category j whilst the other variables, $\delta_k = 0, k = 1, ..., n_c, k \neq j$.

B.2 Independent GPs per category with GP prior, $GP(m, K)$

Given training and test data, $(X, \mathbf{y})$ and $(X^*, \mathbf{y}^*)$ which has n_d continuous covariates for a given level of qualitative factors, perform the following steps,

independent GP model procedure

1. Choose kernel, k which will be used in the Independent-GP model.
2. Hyper-parameters, σ_n^2 (noise variance), $\phi_i, i = 1, ..., n_d$ (length scales for separable kernels), ϕ (length scale for isotropic kernels) and σ_p^2 (process variance), all re-parameterized to a form such as m^2, $e^{-m(i)}$ to ensure the parameters are constrained to be positive. All hyper-parameters are contained in $\boldsymbol{\theta}$. Independent GPs (built for a particular category) contains fewer parameters than the Embedded GP model, hence they do not contain any knowledge of the categories.
3. Use the mean function, $m(\mathbf{x}_i) = h(\mathbf{x}_i)^T\boldsymbol{\beta}$. Prior distribution used for $\boldsymbol{\beta}$ is a Gaussian distribution, $N(\boldsymbol{\beta}|\mathbf{b}, B)$. Note that hyper-parameters, $\boldsymbol{\beta}$ were analytically integrated out (*see* section 2.5) and are hyper-parameters not included in $\boldsymbol{\theta}$. Parameters, $\boldsymbol{\theta}$, are estimated by MAP/ML.

4. Assign initial values to each hyper-parameter in $\boldsymbol{\theta}$ using samples from Gaussian distributions.

5. Use an optimization routine such as, scaled conjugate gradient (SCG) to minimize,
$-\log p(\boldsymbol{\theta}|X, \mathbf{y}) = -1 \times ((-0.5 \times e) - (0.5 \times (\mathbf{y} - \boldsymbol{\mu})' \times (A' \setminus (A \setminus (\mathbf{y} - \boldsymbol{\mu})))) - \frac{Q_{tr}}{2} \times \log(2 \times \pi)) - \log(p(\boldsymbol{\theta}))$ in respect to $\boldsymbol{\theta}$, where $e = \sum_{i=1}^{Q_{tr}} \log \lambda_i$, $\lambda_i = eig(K(X, X) + \mathbf{h}^T B \mathbf{h})$, $A = chol(K(X, X) + \mathbf{h}^T B \mathbf{h})^T$ $\boldsymbol{\mu}$ represents the mean function evaluated at each of the training points. A is an upper triangular matrix has a result of performing Cholesky decomposition on $K(X, X)$.

6. Use the hyper-parameters, σ_n^2, σ_p^2, $\phi_i, i = 1, ..., n_d$ (for separable kernels), ϕ (for isotropic kernels) to evaluate moments of the predictive distribution, $p(y_i^*|\mathbf{y}, X, \mathbf{x}_i^*, \hat{\boldsymbol{\theta}})$, where again Cholesky decomposition is used,

   - $E(y_i^*|\mathbf{y}, X, \mathbf{x}_i^*, \hat{\theta}) = \boldsymbol{\mu}^* + cov(\mathbf{x}_i^*, X) \times (A' \setminus (A \setminus (\mathbf{y} - \boldsymbol{\mu})$,
   - $Var(y_i^*|\mathbf{y}, X, \mathbf{x}_i^*, \hat{\boldsymbol{\theta}}) = cov(\mathbf{x}_i^*, \mathbf{x}_i^*) - cov(\mathbf{x}_i^*, X) \times (A' \setminus (A \setminus cov(X, \mathbf{x}_i^*))$.

   where $\boldsymbol{\mu}^*$ represents the mean function evaluated at each of the test points.

7. Perform steps 1-6 for each combination of qualitative factors (or hence categories).
   Note that,

   - If the minimization of the likelihood, $p(\mathbf{y}|X, \boldsymbol{\theta})$ is required minimize, $-\log p(\mathbf{y}|X, \boldsymbol{\theta}) = -1 \times ((-0.5 \times e) - (0.5 \times (\mathbf{y}' \times (A' \setminus (A \setminus (\mathbf{y})))) - \frac{Q_{tr}}{2} \times \log(2 \times \pi))$ instead of using Step 5.

B.3 Embedded GP (nominal case) for Regression with GP prior, $GP(m, K)$

Embedded GP model for the nominal categorical inputs

Given training and test data, $(X, \mathbf{y})$ and $(X^*, \mathbf{y}^*)$ which has n_d continuous covariates and n_c categories (or n_c different levels of qualitative factors). The following steps are taken for this algorithm,

1. Gather all training and test data points which have the same qualitative factors. Choose kernel, K which will be used in Embedded GP model. The total number of training and test points used in this model is given by Q_{tr} and Q_{te} respectively. The number of qualitative factors used in this algorithm is given by, j_D.

2. Given n_c different combinations of qualitative factors, assign each combination to a hyper-parameter $g_{1,j}$.

   This transforms data input, $\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n_d}, w_{i,j_1}, w_{i,j_2}, ..., w_{i,j_D})' \rightarrow \mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n_d}, v_1, v_2, ..., v_{n_c})'$.
   Vector $\mathbf{v} = (v_1, v_2, ..., v_{n_c}) = (g_{1,1}\delta_1, g_{1,2}\delta_2, ..., g_{1,j}\delta_j, ..., g_{1,n_c}\delta_{n_c})$.

   The transformed input will be used in the Embedded GP. The notation, $\delta_j = 1$ if the data point belongs to category j whilst the other variables, $\delta_k = 0, k = 1, ..., n_c, k \neq j$.

3. Hyper-parameters, σ_n^2 (noise variance), σ_p^2 (process variance), $\phi_i, i = 1, ..., n_d$ (length scales for separable kernels), ϕ (length scale for isotropic kernels) and $g_{1,j}, j = 1, ..., n_c$ all re-parameterized to a form such as m^2, $e^{-m(i)}$ to ensure the parameters are constrained to be positive. These hyper-parameters are contained in $\boldsymbol{\theta}$.

4. Assign an initial value to each hyper-parameter in $\boldsymbol{\theta}$ using samples from Gaussian distributions.

5. Use the mean function, $m(\mathbf{x}_i) = h(\mathbf{x}_i)^T\boldsymbol{\beta}$. The particular form of basis functions considered for data point, $\mathbf{x}_i$, were $h_{2j} = v_j 1$, $h_{2j-1} = v_j x_{i,1}$ $\forall j$, $1 \leq j \leq n_c$. Notice that for a particular input, $\mathbf{x}_i$ the mean function will only contain two terms, yielding a separate mean function per category. Prior distribution used for $\boldsymbol{\beta}$ is a Gaussian distribution, $N(\boldsymbol{\beta}|\mathbf{b}, B)$. Hyper-parameters, $\boldsymbol{\beta}$ are analytically integrated out (*see* section 2.5), so therefore are not included as part of $\boldsymbol{\theta}$.

6. Use an optimization routine such as, scaled conjugate gradient (SCG) to minimize,
$-\log p(\boldsymbol{\theta}|X, \mathbf{y}) = -1\times((-0.5\times e)-(0.5\times(\mathbf{y}-(\boldsymbol{\mu})^T\mathbf{b})'\times(A'\backslash(A\backslash(\mathbf{y}-(\boldsymbol{\mu})^T\mathbf{b})))-\frac{Q_{tr}}{2}\times\log(2\times\pi))-\log(p(\boldsymbol{\theta}))$ in respect to $\boldsymbol{\theta}$, where $e = \sum_{i=1}^{Q_{tr}}\log\lambda_i, \lambda_i = eig(K(X,X)+\mathbf{h}^T B\mathbf{h})$, $A = chol(K(X,X)+\mathbf{h}^T B\mathbf{h})^T$, and $\boldsymbol{\mu}$ is $Q_{tr}\times 1$-dimensional where this is the mean evaluated at each of the training inputs, X. A is an upper triangular matrix has a result of performing Cholesky decomposition on $K(X,X)$, and $\mathbf{h}^T$ is an $Q_{tr}\times n_c$ matrix where each row represents the evaluation of each basis function for a given data point, $\mathbf{x}_i$.

7. Once having obtained $\hat{\boldsymbol{\theta}}$ which is a mode of $p(\boldsymbol{\theta}|X, \mathbf{y})$ use $\hat{\boldsymbol{\theta}}$ to transform the test points such that,

$\mathbf{x}_i^* = (x_{i,1}^*, x_{i,2}^*, ..., x_{i,n_d}^*, w_{i,j_1}^*, w_{i,j_2}^*, ..., w_{i,j_D}^*)' \rightarrow$
$\mathbf{x}_i^* = (x_{i,1}^*, x_{i,2}^*, ..., x_{i,n_d}^*, v_1, v_2, ..., v_{n_c})'$, where vector $\mathbf{v} = (v_1, v_2, ..., v_{n_c}) = (g_{1,1}\delta_1, g_{1,2}\delta_2, ..., g_{1,j}\delta_j, ..., g_{1,n_c}\delta_{n_c})$.
The notation, $\delta_j = 1$ if the data point belongs to category j whilst the other variables, $\delta_k = 0, k = 1, ..., n_c, k \neq j$.

8. Use remaining hyper-parameters, σ_n^2, σ_p^2, $\phi_i, i = 1, ..., n_d$ (for separable kernels), ϕ (for isotropic kernels) to evaluate moments of the predictive distribution, $p(y_i^*|\mathbf{y}, X, \mathbf{x}_i^*, \hat{\boldsymbol{\theta}})$. Note that,

suggestions

   - If the minimization of the likelihood, $p(\mathbf{y}|X, \boldsymbol{\theta})$ is required minimize, do not include $-\log p(\boldsymbol{\theta})$ in Step 6.

B.4 Sampling: Embedded GP (ordinal case) for Regression with GP prior, $GP(0, K)$

Given training and test data, $(X, \mathbf{y})$ and $(X^*, \mathbf{y}^*)$ which has n_d continuous covariates, and n_c categories (or n_c levels of qualitative factors). The number of qualitative factors used in the algorithm is j_D. The following steps are taken for this algorithm,

using the Embedded GP; sampling from posterior/likelihood procedure

1. Gather all training and test data points which have the same qualitative categorical inputs. Choose kernel, k which will be used in Embedded GP. This GP model however contains a total of Q_{tr} training and Q_{te} test points.

2. Given n_c different combinations of qualitative factors, assign each combination to a hyper-parameter $g_{1,j}$.

   This transforms data input,

   $$\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n_d}, w_{i,j_1}, w_{i,j_2}, ..., w_{i,j_D})' \rightarrow \mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n_d}, g_{1,j})'.$$

   The transformed input will be used in the Embedded GP.

3. Hyper-parameters, σ_n^2 (noise variance), σ_p^2 (process variance), $\phi_i, i = 1, ..., n_d$ (length scales for separable kernels), ϕ (length scale for isotropic kernels) and $g_{1,j}, j = 1, ..., n_c$ all re-parameterized to a form such as m^2, $e^{-m(i)}$ to ensure the parameters are constrained to be positive. These hyper-parameters are contained in $\boldsymbol{\theta}$.

4. Choose initial values for $\boldsymbol{\theta}$ using designs such as sampling from Gaussian distributions.

5. Use an sampling routine to obtain samples from,

   $p(\boldsymbol{\theta}|X, \mathbf{y}) = \exp^{-(-1\times((-0.5\times e)-(0.5\times(\mathbf{y}'\times(A'\setminus(A\setminus(\mathbf{y}))))-\frac{Q_{tr}}{2}\times\log(2\times\pi))-\log(p(\boldsymbol{\theta}))))}$ where $e = \sum_{i=1}^{Q_{tr}} \log \lambda_i$, $\lambda_i = eig(K(X, X))$, $A = chol(K(X, X))'$. A is an upper triangular matrix has a result of performing Cholesky decomposition on $K(X, X)$. Prior distribution over the hyper-parameters is indicated by $p(\boldsymbol{\theta})$ which is factorizable over each hyper-parameter in $\boldsymbol{\theta}$.

6. Having obtained i_c samples, $\boldsymbol{\theta}^{(1)}, ..., \boldsymbol{\theta}^{(i_c)} \sim p(\boldsymbol{\theta}|X, \mathbf{y})$.

7. Consider a single test input data point, $\mathbf{x}_j^*$. For $i = 1, ..., i_c$, transform the test point such that,

   $$\mathbf{x}_k^* = (x_{k,1}^*, x_{k,2}^*, ..., x_{k,n_d}^*, w_{k,j_1}^*, w_{k,j_2}^*, ..., w_{k,j_D}^*)' \rightarrow \mathbf{x}_k^* = (x_{k,1}, x_{k,2}, ..., x_{k,n_d}, g_{1,j}^{(i)})'.$$

   Compute this over all test points.

8. Use remaining hyper-parameters, σ_n^2, σ_p^2, $\phi_i, i = 1, ..., n_d$ (for separable kernels), ϕ (for isotropic kernels) from current sample, $\boldsymbol{\theta}^{(i)}$ to evaluate moments of the predictive distribution, $p(y_j^*|\mathbf{y}, X, \mathbf{x}_j^*, \boldsymbol{\theta}^{(i)})$, (where again Cholesky decomposition is used),

   - $E(y_j^*|\mathbf{y}, X, \mathbf{x}_j^*, \boldsymbol{\theta}^{(i)}) = cov(\mathbf{x}_j^*, X) \times (A' \setminus (A \setminus (\mathbf{y}')))$,
   - $Var(y_j^*|\mathbf{y}, X, \mathbf{x}_j^*, \boldsymbol{\theta}^{(i)}) = cov(\mathbf{x}_j^*, \mathbf{x}_j^*) - cov(\mathbf{x}_i^*, X) \times (A' \setminus (A \setminus (cov(X, \mathbf{x}_i^*))))$.

9. Having performed step 9 for all i_c samples, use the results from these steps to evaluate (where $\boldsymbol{\theta}$ is integrated over),

   - $E(y_j^*|\mathbf{x}_j^*, X, \mathbf{y}) = \frac{1}{i_c} \sum_{i=1}^{i_c} E(y_j^*|\mathbf{x}_j^*, X, \mathbf{y}, \boldsymbol{\theta}^{(i)})$
   - $Var(y_j^*|\mathbf{x}_j^*, X, \mathbf{y}) = \frac{1}{i_c} \sum_{i=1}^{i_c} Var(y_j^*|\mathbf{x}_j^*, X, \mathbf{y}, \boldsymbol{\theta}^{(i)}) + Var(E(y_j^*|\mathbf{y}, X, \mathbf{x}_j^*, \boldsymbol{\theta}^{(i)}))$.

10. Perform steps 10-11 for all test data inputs, to obtain the moments of predictive distribution, $p(y_i^*|\mathbf{x}_i^*, X, \mathbf{y})$. Note that, suggestions

    - If sampling from the likelihood, $p(\mathbf{y}|X, \boldsymbol{\theta})$ is required use $-\log p(\mathbf{y}|X, \boldsymbol{\theta}) = -1 \times ((-0.5 \times e) - (0.5 \times (\mathbf{y}' \times (A' \setminus (A \setminus (\mathbf{y})))) - \frac{Q_{tr}}{2} \times \log(2 \times \pi))$ instead of using Step 5.

- in the case of nominal case, transform data input $\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n_d}, w_{i,j_1}, w_{i,j_2}, ..., w_{i,j_D})' \rightarrow$ $\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n_d}, g_{1,1}\delta_1, g_{1,2}\delta_2, ..., g_{1,j}\delta_j, ..., g_{1,n_c}\delta_{n_c})'$ in place of Step 2 and replace Step 7 by, $\mathbf{x}_k^* = (x_{k,1}^*, x_{k,2}^*, ..., x_{k,n_d}^*, w_{k,j_1}^*, w_{k,j_2}^*, ..., w_{k,j_D}^*)' \rightarrow$ $\mathbf{x}_k^* = (x_{k,1}^*, x_{k,2}^*, ..., x_{k,n_d}^*, g_{1,1}^{(i)}\delta_1, g_{1,2}^{(i)}\delta_2, ..., g_{1,j}^{(i)}\delta_j, ..., g_{1,n_c}^{(i)}\delta_{n_c})'$.

B.5 Hypersphere GP for Regression with GP prior, $GP(0, K)$

reasons for building our implementation of Hypersphere GP model

Because, Zhou et al. (2010) does not provide the predictive variances for the Hypersphere GP model (in MATLAB code), the decision was made to implement our version of the model in MATLAB. Our implementation of the Hypersphere GP model is shown below,

Given training and test data, $(X, \mathbf{y})$ and $(X^*, \mathbf{y}^*)$ which has n_d continuous co-variates and n_c categories (or n_c levels of qualitative factors). The following steps are taken for this algorithm,

1. Gather all training and test data points which have the same qualitative categorical inputs. Choose kernel, k which will be used in the Hypersphere GP model. The total amount of training and test points used in this Hypersphere GP model is given by Q_{tr} and Q_{te} respectively.
2. Given n_c different combinations of qualitative factors, assign each combination to value j, where $j = 1, ..., n_c$, which will be referred to as categories. This variable is not being inferred.
3. For each pair of categories, i, and j, generate variables $l_{i,j}, \forall i < j$.
4. Hyper-parameters, σ_n^2 (noise variance), σ_p^2 (process variance), $\phi_i, i = 1, ..., n_d$ (length scales for separable kernels), ϕ (length scale for isotropic kernels) $l_{i,j}$, are all re-parameterized to a form such as m^2, $e^{-m(i)}$ to ensure the parameters are constrained to be positive. These hyper-parameters are contained in $\boldsymbol{\theta}$.
5. In addition, all variables, $l_{i,j}$ have to be constrained to lie in interval $[0, \pi]$. This is achieved by using the following parameterization $(\arctan m + \frac{\pi}{2})$ for all $l_{i,j}\ \forall i \leq j$ where $i < j$.
6. Assign initial values to each of the hyper-parameters, $\boldsymbol{\theta}$ using a Latin Hypercube design or by sampling from Gaussian distributions. Construct the correlation matrix, T.
7. Use an optimization routine such as, scaled conjugate gradient (SCG) to minimize, $-\log p(\boldsymbol{\theta}|X, \mathbf{y}) = -1 \times ((-0.5 \times e) - (0.5 \times (\mathbf{y}' \times (A' \setminus (A \setminus (\mathbf{y}))))) - \frac{Q_{tr}}{2} \times \log(2 \times \pi)) - \log(p(\boldsymbol{\theta}))$

   in respect to $\boldsymbol{\theta}$, where $e = \sum_{i=1}^{Q_{tr}} \log \lambda_i, \lambda_i = eig(K(X, X))$, $a = chol(K(X, X))'$. A is an upper triangular matrix has a result of performing Cholesky decomposition on $K(X, X)$.
8. Once having obtained $\hat{\boldsymbol{\theta}}$ which MAP, use these hyper-parameters, σ_n^2, σ_p^2, $\phi_i, i = 1, ..., n_d$ (for separable kernels), ϕ (for isotropic kernels) and $l_{i,j}\ \forall i \leq j$ to

evaluate moments of the predictive distribution, $p(y_i^*|\mathbf{y}, X, \mathbf{x}_i^*, \hat{\boldsymbol{\theta}})$, (where again Cholesky decomposition is used)

- $E(y_i^*|\mathbf{y}, X, \mathbf{x}_i^*, \hat{\boldsymbol{\theta}}) = cov(\mathbf{x}_i^*, X) \times (A' \setminus (A \setminus (\mathbf{y}')))$,
- $Var(y_i^*|\mathbf{y}, X, \mathbf{x}_i^*, \hat{\boldsymbol{\theta}}) = cov(\mathbf{x}_i^*, \mathbf{x}_i^*) - cov(\mathbf{x}_i^*, X) \times (A' \setminus (A \setminus (cov(X, \mathbf{x}_i^*))))$.

Note that, suggestions

- when the Hypersphere GP model uses a mean function, it will be in the form, $m(\mathbf{x}_i) = h(\mathbf{x}_i)^T \boldsymbol{\beta}$. The basis functions considered for data point, $\mathbf{x}_i$ were $h_{2j} = v_j 1$, and $h_{2j-1} = v_j x_{i,1}$ for $\forall j,\ 1 \leq j \leq n_c$. Normal priors were used on $\boldsymbol{\beta}$, where $N(\boldsymbol{\beta}|\mathbf{b}, B)$. These parameters, $\boldsymbol{\beta}$, were integrated out analytically. These parameters are not included in $\boldsymbol{\theta}$. Instead of computing steps $7-8$, steps 6 and 8 in Appendix B.3 are computed instead, with the notable difference in the hyper-parameters, $\boldsymbol{\theta}$ used.

Appendix C

Simulated Tempering Suggestion

This section reviews the modified version of Simulated Tempering (ST) which was used for experimentation in Chapter 5.

suggestions for the Simulated Tempering algorithm

Suppose the distribution to be sampled from is $p(\boldsymbol{\theta})$. Stationary distributions at each temperature level, i, on the temperature ladder, would have the form,

$$\frac{p(\boldsymbol{\theta})}{T_i} = \exp^{-(-\frac{\log p(\boldsymbol{\theta})}{T_i})}.$$

huge mode; possible numerical issues

A possible mode, $\boldsymbol{\beta}_0$ for distribution, $p(\boldsymbol{\theta})$ could take a very big value, and programs such as MATLAB could struggle with values such as $\exp^{-(-p(\boldsymbol{\beta}_0))}$ in this case due to numerical reasons. To make ST more stable, the following ST algorithm has been suggested and used in our work.

our implementation of ST algorithm

1. Use (non-linear) optimization routine such as SCG to find the maximum of the distribution, $\log p(\boldsymbol{\theta})$. Use the (global) mode of $p(\boldsymbol{\theta})$ found to re-scale the stationary distributions, $\pi(\boldsymbol{\theta}, T_i)$. Stationary distributions will take the form,

$$\pi_i(\boldsymbol{\theta}, T_i) = \exp^{-(\frac{-\log p(\boldsymbol{\theta}) + \log(p(\boldsymbol{\beta}_0))}{T_i})}.$$

2. Choose the temperatures T_i on the temperature ladder of size m, and obtain the normalization constants, Z_i for $i = 1, ..., m$. The m temperature levels are indicated by $i = 1, ..., m$. Normalization constants are obtained using importance sampling, where the sampling distribution is a product of uni-variate Gaussian distribution.

3. Begin the ST algorithm at the highest temperature in the temperature ladder. Initialize values for $\boldsymbol{\theta}$.

4. At the current temperature, update $\boldsymbol{\theta}$ by using component-wise MH updates using the distribution, $\pi(\boldsymbol{\theta}, T_i)$. This is done for 6 iterations.

5. Propose increasing or decreasing the temperature, T_i by one temperature level, $j = i \pm 1$, according to temperature transition probabilities, $p_{1,2} = 1.0$ if $i = 1$, $p_{m,m-1}$ if $i = m$ and $p_{i,i+1} = p_{i,i-1} = 0.5$ if $1 < i < m$.

6. MH-step is used to decide whether to accept or reject the proposed temperature, T_j based on,

$$\alpha = \frac{Z_j \pi_j(\boldsymbol{\theta}, T_j) p_{j,i}}{Z_i \pi_i(\boldsymbol{\theta}, T_i) p_{i,j}},$$

where the temperature change from i-th to j-th positions in the temperature ladder is given with probability $\min(1, \alpha)$.

7. Repeat steps 3-5 for a M number of temperature transitions.

8. Keep the samples which correspond to the temperature $T_1 = 1$.

The normalization constant, Z_i for the lowest temperature $T_1 = 1$ on the ladder, could be very large because of mode, β_0, gives a high functional value, $p(\beta_0)$. By rescaling the stationary distributions as like this, will make the ST algorithm more stable. This version of ST will be used for experimentation.

scaling down; factor for the normalization constants

Bibliography

Bastos, L. S. (2010), Validating Gaussian process models in computer experiments, University of Sheffeld, Technical report.

Behrens, G., Friel, N. & Hurn, M. (2010), *Tuning Tempered Transitions*, ArXiv e-prints.

Bertsimas, D. & Tsitsiklis, J. (1993), 'Simulated annealing', *Statistical Science* **8**.

Bishop, C. M. (2006), *Pattern Recognition and Machine Learning*, Springer Science, LLC, London.

Boukouvalas, A. (2010), Emulation of random output simulators, Technical report.

Broderick, T. & Gramacy, R. B. (2010), 'Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with tgp Version 2, an R Package for Treed Gaussian Process Models', *Journal of Statistical Software* **33**(6), 1–48.

Brooks, S. P. & Gelman, A. (1998), 'General methods for monitoring convergence of iterative simulations', *J. Comput. Graph. Stat. 7* **7**, 434–455.

Brouwer, R. K. (2002), 'A feed-forward network for input that is both categorical and quantitative', *Neural Networks* **15**, 881–890.

Chao, D. L., Halloran, M. E., Obenchain, V. J. & Longini, I. M. (2010), 'Flute, a publicly available stochastic influenza epidemic simulation model', **6**(1).

Chipman, H. A., George, E. I. & McCulloch, R. E. (2002), 'Bayesian Treed Models', *Machine Learning* **48**, 303–324.

Christen, J. A. & Fox, C. (2005), 'MCMC using an approximation', *Journal of Computer and Graphical Statistics* **14**(4), 795–810.

Chu, W. & Ghahramani, Z. (2005), 'Gaussian Processes for Ordinal Regression', *Journal of Machine Learning Research* **6**, 1019–1041.

Cowles, M. K., Roberts, G. O. & Rosenthal, J. S. (1999), 'Possible biases induced by MCMC convergence diagnostics', *J. Stat. Comp. Sim* **64**, 87–104.

Feng, C. X. (2006), 'Selection and validation of predictive regression and neural network model based on designed experiments.', *IIE Transactions* **38**, 13–23.

Frank, A. & Asuncion, A. (2010), 'UCI machine learning repository'.
URL: *http://archive.ics.uci.edu/ml*

Gelman, A. & Shirley, K. (2010), Inference from simulations and monitoring convergence, *in* 'Handbook of Markov Chain Monte Carlo: Methods and Applications'.

Gramacy, R. B. & Lee, H. K. H. (2008), 'Bayesian treed Gaussian process models with an application to computer modeling', *Journal of the American Statistical Association* **103**, 1119–1130.

Hannah, L. A., Blei, D. M. & Powell, W. B. (2011), 'Dirichlet process mixtures of generalized linear models', *Journal of Machine Learning Research* **1**, 1–33.

Johnson, A. A., Jones, G. L. & Neath, R. C. (2011), Component-wise Markov chain Monte Carlo: Uniform and geometric ergodicity under mixing and composition, Technical report.

Kim, H. & Ghahramani, Z. (2004), The EM-EP algorithm for Gaussian process classification, Technical report.

Lauwerrs, L., Barbe, K., Van Moer, W. & Pintelon, R. (2009), ' Estimating the parameters of a Rice distribution: a Bayesian approach', *Instrumentation and Measurement Technology Conf. (Singapore)* **114-7**, 5–7.

Lawrence, N. D. (2005), 'Probabilistic non-linear principal component analysis with Gaussian process latent variable models', *Journal of Machine Learning Research* **6**.

Li, Y. & Protopopescu, V. (2004), 'Accelerated simulated tempering', *Physics letters A 2004* **328**, 274–283.

Nabney, I. T. (2002), *NETLAB-Algorithms for Pattern Recognition*, Springer-Verlag.

Nickisch, H. & Rasmussen, C. E. (2008), 'Approximations for binary Gaussian process classification', *J. Mach. Learn. Res* **9**, 2035–2078.

Perrin, G., Descombes, X. & Zerubia, J. (2005), 'Adaptive simulated annealing for energy minimization problem in a marked point process application'.

Posada, D. & Buckley, T. R. (2004), 'Model selection and model averaging in phylogenetics: advantages of Akaike information criterion and Bayesian approaches over likelihood ratio tests', *Syst Biol* **53**, 793–808.

Qian, P. Z. G., Wu, H. & Wu, C. F. J. (2008), 'Gaussian process models for computer experiments with qualitative and quantitative factors Technometrics', **50**(3), 383–396.

Raftery, A. E. (1993), Bayesian model selection in structural equation models, Technical report.

Rasmussen, C. E. (2006), Gaussian processes for machine learning, MIT Press, Cambridge.

Snelson, E., Rasmussen, C. E. & Ghahramani, Z. (2004), 'Warped gaussian processes', **16**.

Snelson, E. et al. (2011), 'Documentation for gpml matlab code version 3.1', `http://www.gaussianprocess.org/gpml/code/matlab/doc/index.html`.

Taddy, M., Gramacy, R. & Polson, N. (2010), 'Dynamic trees for learning and design', *Tech. Rep. arXiv:0912.1636* **1636**.

Titsias, M., Lawrence, N. D. & Rattray, M. (2009), Efficient sampling for Gaussian process inference using control variables, *in* 'Advances in Neural Information Processing Systems 21', pp. 1681–1688.

Venna., J. (2007), 'Dimensionality reduction for visual exploration of similarity structures. phd thesis'.

Walsh, B. (2004), 'Markov Chain Monte Carlo and Gibbs Sampling'.
URL: *http://web.mit.edu/wingated/www/introductions/mcmc-gibbs-intro.pdf*

Wang, C.-H. (1988), Height-diameter equations for sixteen tree species in the central western willamette valley of oregon, Technical report, Oregon State University.

Zhou, Q., Qian, P. Z. G., Wu, H. & Zhou, S. (2010), A simple approach to emulation for computer models with qualitative and quantitative factors, Technical report, University of Wisconsin.