Multivariate Stochastic Process Models for Correlated Responses of Mixed Type

Tony Pourmohamad
Department of Applied Mathematics and Statistics,
University of California, Santa Cruz
and
Herbert H.K. Lee
Department of Applied Mathematics and Statistics,
University of California, Santa Cruz

September 20, 2015

Abstract

We propose a new model for correlated outputs of mixed type, such as continuous and binary outputs, with a particular focus on joint regression and classification, motivated by an application in constrained optimization for computer simulation modeling. Our framework is based upon multivariate stochastic processes, extending Gaussian process methodology for modeling of continuous multivariate spatial outputs by adding a latent process structure that allows for joint modeling of a variety of types of correlated outputs. In addition, we implement fully Bayesian inference using particle learning, which allows us to conduct fast sequential inference. We demonstrate the effectiveness of our proposed methods on both synthetic examples and a real world hydrology computer experiment optimization problem where it is helpful to model the black box objective function as correlated with satisfaction of the constraint.

Keywords: Gaussian process, particle learning, Bayesian statistics, constrained optimization, computer simulation experiment

1 Introduction

The problems of regression and classification are both well-studied individually, but there has been limited work on the problem of combined regression and classification when these outputs are correlated, particularly in the nonparametric setting. Only recently has the literature moved beyond traditional parametric assumptions. We are motivated by the problem of constrained optimization where both the objective function and the constraints are unknown and potentially expensive to evaluate. Thus we seek an efficient statistical model to serve as a fast approximation to the true objective function and constraints, which in our applications are computer simulation experiments. In the case that the simulator only returns whether a constraint is satisfied, and not any measure of distance to satisfaction, we need to jointly model a continuous objective function and one or more binary constraints. Constrained optimization is typically difficult because at least one of the constraints operates in opposition to the objective function, i.e., they are negatively correlated. We propose here a nonparametric model to jointly model continuous and binary outputs, and the framework is flexible enough to include a wide variety of other types of outputs. Our approach builds upon the standard computer emulation approach in the literature of using Gaussian process (GP) models (Santner et al., 2003).

Joint modeling of outputs of different types, also called multiway or mixed type responses, can be a difficult task. When the outputs are known or suspected to be correlated, it is common practice to use latent processes to induce correlation between them (Sammel et al., 1997; Moustaki and Knott, 2000). However, most of these latent methods rely on either simple linear models or restrictive parametric assumptions. Recent papers have started to utilize most robust nonparametric models, such as infinite mixtures of linear models (Zhe et al., 2015). Typically in computer simulation experiments, constrained optimization is very challenging because the outputs of the simulators arise from highly nonlinear functions. This lack of linearity is what makes nonparametric methods so desirable. An active area of research in machine learning, multi-task learning builds predictive models based on the learning of multiple tasks, in our case learning mixed type outputs, at the same time. The performance of multi-task learning methods is highly dependent upon the sharing of information, or induced correlation, across each task. A nonparametric

extension, Yu et al. (2005), used multi-task GPs for sharing information across multiple tasks of the same type. However, the growing literature on multi-task GPs (Bonilla et al., 2008; Hayashi et al., 2012) does not seem to suggest that the model can facilitate the modeling of outputs of mixed types. Similar to the approach we will take in this paper, Liu et al. (2013) makes an attempt to address this problem, however, a major limitation of that work is that they only consider the case of two correlated outputs. We propose here a flexible nonparametric model capable of handling $p \geq 2$ correlated outputs to address this gap. The Gaussian process framework has proved to be an effective tool for modeling both regression and classification (Neal, 1999). Multivariate regression GPs (Wackernagel, 2003) provide a basis for modeling correlated outputs. We build upon these ideas to create a new GP-based model for correlated outputs of mixed type, where each output uses a transformation function to map back to the regression setting. Chan (2013) explored this same idea, but was limited by only considering correlated outputs of the same type and by assuming that the likelihood function takes the generic form of the multivariate exponential family distribution. We bypass these limitations by allowing for a more general likelihood function and create a fully generalized family of models that utilize standard link functions, including but not limited to the identity for regression and the logistic for classification, but more general links also fit into our framework. Another similar approach, Xu et al. (2012) and Zhe et al. (2013) utilized latent tensor-valued Gaussian process models to model mixed type outputs from a Bayesian point of view, however, both works employed only variational techniques for inference. Another key innovation of our work is fully Bayesian inference, through particle learning, whereas Liu et al. (2013), Xu et al. (2012), Chan (2013) and Zhe et al. (2013) provide for only approximate Bayesian inference.

We take a fully Bayesian approach, which can require significant computational effort. To address this concern, we build upon recent work on sequential Monte Carlo methods for GPs (Gramacy and Polson, 2011; Montagna and Tokdar, 2013). Particle learning allows for fast inference, and also allows for sequential inference when the data arrive over time, as is the case in computer experiments. As a special case, we believe this is the first implementation of particle learning for a multivariate regression GP.

The remainder of this paper is organized as follows. In the next section, we review

the separable multivariate Gaussian process and setup the necessary modeling framework for the rest of the paper. Section 3 introduces our novel joint regression and classification model. We review the sequential Monte Carlo technique, particle learning, in Section 4, and explain how fast sequential inference can be conducted on our joint regression and classification model with an extension to a similar stochastic process model. Section 5 demonstrates the applicability of the models presented with a number of illustrative examples and comparisons with previous work. Section 6 concludes with some discussion.

2 Multivariate Gaussian Process

We consider a stochastic process returning a p-dimensional output $\mathbf{y} \in \mathbb{R}^p$ for a given d-dimensional input $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$. We think of the stochastic process as a function $\mathbf{f} : \mathcal{X} \to \mathbb{R}^p$ for some (possibly high dimensional) input space $\mathcal{X}$. Similar to Conti and O'Hagan (2010) and Fricker et al. (2013), from a Bayesian perspective, we regard $\mathbf{f}(\cdot)$ as an unknown function and represent the uncertainty surrounding it through the use of the p-dimensional multivariate Gaussian process

$$\mathbf{f}(\cdot) \sim \mathrm{GP}_p\left(\boldsymbol{\mu}(\cdot), \mathbf{C}(\cdot,\cdot)\right), \tag{1}$$

where $\boldsymbol{\mu}$ is a mean function and $\mathbf{C}$ is a covariance function. The existence of the multivariate Gaussian process depends on the specification of a valid cross-covariance function $\mathbf{C}(\mathbf{x},\mathbf{x}') = \mathrm{Cov}(\mathbf{f}(\mathbf{x}),\mathbf{f}(\mathbf{x}'))$ for $\mathbf{x},\mathbf{x}' \in \mathcal{X}$ (Wackernagel, 2003). Generating valid, as well as tractable, cross-covariance functions is not a simple task. Many methods have been proposed, such as: separable models (Mardia and Goodall, 1993; Banerjee and Gelfand, 2002), convolution of covariance functions (Gaspari and Cohn, 1999; Majumdar and Gelfand, 2007), and the linear model of coregionalization (Goulard and Voltz, 1992; Wackernagel, 2003; Gelfand et al., 2004). The approaches in this paper work for more general covariance structures beyond the separable model, for example we have tried them with the linear model of coregionalization, but we focus on the separable model as we find it works well in practical applications, providing sufficient flexibility without too much additional computational expense. In the section that follows, we briefly discuss separable models.

2.1 Separable Model

One of the simplest ways of achieving a valid cross-covariance function is to take a valid univariate correlation function $\rho(\mathbf{x}, \mathbf{x}')$ and a valid $p \times p$ positive semidefinite matrix $\mathbf{T}$ so that

$$\mathbf{C}(\mathbf{x}, \mathbf{x}') = \rho(\mathbf{x}, \mathbf{x}')\mathbf{T}. \tag{2}$$

The cross covariance function in (2) is said to be a separable model. Letting $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)$ be the collection of all inputs observed so far in $\mathcal{X}$, the resulting $np \times np$ covariance matrix for $\mathbf{Y}^T = (\mathbf{y}_1^T, ..., \mathbf{y}_n^T)$, where $\mathbf{y}_i^T = (f_i(\mathbf{x}_1), ..., f_i(\mathbf{x}_n))$, is

$$\mathbf{C}(\mathbf{X}, \mathbf{X}) = \mathbf{R} \otimes \mathbf{T} = \begin{pmatrix} \rho(\mathbf{x}_1, \mathbf{x}_1)\mathbf{T} & \cdots & \rho(\mathbf{x}_1, \mathbf{x}_n)\mathbf{T} \\ \vdots & \vdots & \vdots \\ \rho(\mathbf{x}_n, \mathbf{x}_1)\mathbf{T} & \cdots & \rho(\mathbf{x}_n, \mathbf{x}_n)\mathbf{T} \end{pmatrix}, \tag{3}$$

where we denote $\mathbf{\Sigma} = \mathbf{C}(\mathbf{X}, \mathbf{X})$ and $\mathbf{R}$ is the $n \times n$ correlation matrix with $\mathbf{R}_{ij} = \rho(\mathbf{x}_i, \mathbf{x}_j)$. Clearly, $\mathbf{\Sigma}$ is positive semidefinite since $\mathbf{R}$ and $\mathbf{T}$ are. There are some clear advantages to using a separable model, for instance, $|\mathbf{\Sigma}| = |\mathbf{R}|^p |\mathbf{T}|^n$ and $\mathbf{\Sigma}^{-1} = \mathbf{R}^{-1} \otimes \mathbf{T}^{-1}$ which means that working with $\mathbf{\Sigma}$ requires working with a $p \times p$ and $n \times n$ matrix instead of a $np \times np$ matrix. Additionally, from a Bayesian perspective, using a separable model allows for placing a conjugate prior on $\mathbf{\Sigma}$ (Banerjee et al., 2004). Conti and O'Hagan (2010) placed an improper inverse-Wishart prior on $\mathbf{\Sigma}$, which leads to a proper inverse-Wishart posterior for $\mathbf{\Sigma}$ and allows for $\mathbf{\Sigma}$ to be analytically integrated out of the posterior predictive process.

2.2 Model Building and Prediction

We treat the unknown function $\mathbf{f}(\cdot)$ as a multivariate stochastic process and model $\mathbf{f}(\cdot)$ as

$$\begin{aligned} \mathbf{f}(\cdot) &= \boldsymbol{\mu}(\cdot) + \boldsymbol{\omega}(\cdot) \\ \boldsymbol{\mu}(\cdot) &= (\mathbf{I}_p \otimes \mathbf{H}) \operatorname{vec}(\mathbf{B}) \\ \boldsymbol{\omega}(\cdot) | \mathbf{T}, \boldsymbol{\phi}, \eta &\sim \mathrm{GP}_p(\mathbf{0}, \mathbf{C}(\cdot, \cdot)), \end{aligned} \tag{4}$$

where vec$(\cdot)$ is the "vec" operator which stacks the columns of its matrix argument from left to right into a single vector. Here, $\mathbf{I}_p$ denotes the $p \times p$ identity matrix, $\mathbf{H}^T =$

$[\mathbf{h}(\mathbf{x}_1)\cdots\mathbf{h}(\mathbf{x}_n)] \in \mathbb{R}^{q\times n}$ is a matrix of regression functions, $\mathbf{B} = [\boldsymbol{\beta}_1\cdots\boldsymbol{\beta}_p] \in \mathbb{R}^{q\times p}$ is a matrix of regression coefficients and the matrix valued covariance function, $\mathbf{C}(\cdot,\cdot)$, depends on covariance parameters $\mathbf{T}$ and $\boldsymbol{\psi} = \{\boldsymbol{\phi}, \eta\}$ where $\boldsymbol{\psi}$ represents parameters governing the correlation function ρ, which we take in this paper to be the length-scale parameter $\boldsymbol{\phi}$ and nugget parameter η. The length-scale parameter, $\boldsymbol{\phi}$, plays the role of determining how fast the spatial correlation decays throughout the input space, while the nugget, η, is considered as random noise and typically represents measurement error or short scale variability (Cressie, 1993; Diggle and Ribeiro Jr., 2007) in the Gaussian process. Thus, the nugget provides a mechanism for introducing measurement error into the Gaussian process. Assuming separability of the covariance function in (4) allows us to write the likelihood for the data as the following matrix Normal distribution

$$\mathbf{D}|\mathbf{B},\mathbf{T},\boldsymbol{\phi},\eta \sim \mathrm{N}_{n,p}\left(\mathbf{HB},\mathbf{R},\mathbf{T}\right), \tag{5}$$

(Rowe, 2003) where we arrange the data vector $\mathbf{Y}$ into the output matrix $\mathbf{D}$ such that $\mathrm{vec}(\mathbf{D}) = \mathbf{Y}$. From a Bayesian point of view, all that is left is to place prior distributions on the unknown parameters of the model and to update the posterior distribution of the unknown parameters via Bayes' theorem. Lacking strong prior information for $\mathbf{B}$ and $\mathbf{T}$, we follow Conti and O'Hagan (2010) and place the following joint improper prior for $\mathbf{B}$ and $\mathbf{T}$

$$p(\mathbf{B},\mathbf{T}|\boldsymbol{\psi}) \propto |\mathbf{T}|^{-(p+1)/2}. \tag{6}$$

Specifying an arbitrary choice of prior $p(\boldsymbol{\psi})$ for $\boldsymbol{\psi}$ we obtain the posterior distribution

$$\begin{aligned} p(\mathbf{B},\mathbf{T},\boldsymbol{\psi}|\mathbf{D}) &\propto |\mathbf{R}|^{-p/2}|\mathbf{T}|^{-(n-q+p+1)/2}p(\boldsymbol{\psi}) \\ &\times \exp\left\{-\frac{1}{2}\left[\mathrm{tr}\left(\mathbf{D}^T\mathbf{G}\mathbf{D}\mathbf{T}^{-1}\right) + \mathrm{tr}\left(\left(\mathbf{B}-\hat{\mathbf{B}}\right)^T\left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)\left(\mathbf{B}-\hat{\mathbf{B}}\right)\mathbf{T}^{-1}\right)\right]\right\}, \end{aligned} \tag{7}$$

where $\mathbf{G} = \mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{H}\left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{R}^{-1}$ and $\hat{\mathbf{B}} = \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{D}$ is the generalized least squares estimator of $\mathbf{B}$. Our choice of prior in (6) allows us to integrate out $\mathbf{B}$ and $\mathbf{T}$ from the above posterior distribution (7) resulting in the marginal posterior distribution

$$p(\boldsymbol{\psi}|\mathbf{D}) \propto |\mathbf{R}|^{-p/2}|\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}|^{-p/2}|\mathbf{D}^T\mathbf{G}\mathbf{D}|^{-(n-q)/2}p(\boldsymbol{\psi}). \tag{8}$$

Eliciting prior distributions for the correlation parameters $\boldsymbol{\psi}$ is, in general, a difficult task. We enforce the caveat that proper priors must be placed on the correlation parameters $\boldsymbol{\phi}$ and nugget η in order to ensure a proper marginal posterior distribution. In either case, Monte Carlo methods will need to be used in order to obtain posterior samples of $\boldsymbol{\psi}$.

Of main concern is deriving the posterior distribution of $\mathbf{f}(\cdot)$ given the output data $\mathbf{D}$ since this distribution will allow us to make predictions, and quantify our uncertainties, for outputs at new inputs $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, ..., \tilde{\mathbf{x}}_m)$. Conditional on $\mathbf{B}, \mathbf{T}, \boldsymbol{\psi}$, and $\mathbf{D}$ the posterior distribution of $\mathbf{f}(\cdot)$ is

$$\mathbf{f}(\cdot)|\mathbf{B}, \mathbf{T}, \boldsymbol{\psi}, \mathbf{D} \sim \mathrm{GP}_{mp}\left(\mathrm{vec}(\boldsymbol{\mu}^*(\cdot)), \mathbf{T} \otimes \mathbf{C}^*(\cdot, \cdot)\right) \tag{9}$$

where for $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, ..., \tilde{\mathbf{x}}_m) \in \mathcal{X}$

$$\boldsymbol{\mu}^*(\tilde{\mathbf{X}}) = \tilde{\mathbf{H}}\mathbf{B} + \mathbf{F}\mathbf{R}^{-1}\left(\mathbf{D} - \mathbf{H}\hat{\mathbf{B}}\right), \tag{10}$$

$$\mathbf{C}^*(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) = \mathbf{C}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) - \mathbf{F}\mathbf{R}^{-1}\mathbf{F}^T, \tag{11}$$

and $\tilde{\mathbf{H}}^T = [\mathbf{h}(\tilde{\mathbf{x}}_1), ..., \mathbf{h}(\tilde{\mathbf{x}}_m)] \in \mathbb{R}^{q \times m}$ is a matrix of regression functions and $\mathbf{F} = \rho(\tilde{\mathbf{X}}, \mathbf{X}) \in \mathbb{R}^{m \times n}$. Typically, integrating the correlation parameters $\boldsymbol{\psi}$ out of (9) cannot be done analytically and so one instead works with the posterior distribution of $\mathbf{f}(\cdot)$ conditional on the output data $\mathbf{D}$ and correlation parameters $\boldsymbol{\psi}$. Integrating $\mathbf{B}$ and $\mathbf{T}$ out of (9) yields the multivariate $\mathcal{T}$ process (Gupta and Nagar, 2000)

$$\mathbf{f}(\cdot)|\boldsymbol{\psi}, \mathbf{D} \sim \mathcal{T}_{mp}\left(\mathrm{vec}(\boldsymbol{\mu}^*(\cdot)), \hat{\mathbf{T}} \otimes \mathbf{C}^*(\cdot, \cdot), n - q\right) \tag{12}$$

with $n - q$ degrees of freedom, where for $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, ..., \tilde{\mathbf{x}}_m) \in \mathcal{X}$

$$\boldsymbol{\mu}^*(\tilde{\mathbf{X}}) = \tilde{\mathbf{H}}\hat{\mathbf{B}} + \mathbf{F}\mathbf{R}^{-1}\left(\mathbf{D} - \mathbf{H}\hat{\mathbf{B}}\right), \tag{13}$$

$$\begin{aligned}\mathbf{C}^*(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) = \mathbf{C}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) - \mathbf{F}\mathbf{R}^{-1}\mathbf{F}^T \\ + \left(\tilde{\mathbf{H}} - \mathbf{F}\mathbf{R}^{-1}\mathbf{H}\right)\left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\left(\tilde{\mathbf{H}} - \mathbf{F}\mathbf{R}^{-1}\mathbf{H}\right)^T,\end{aligned} \tag{14}$$

and $\hat{\mathbf{T}} = (n - q)^{-1}\left(\mathbf{D} - \mathbf{H}\hat{\mathbf{B}}\right)^T \mathbf{R}^{-1}\left(\mathbf{D} - \mathbf{H}\hat{\mathbf{B}}\right)$ denotes the generalized least squares estimator of $\mathbf{T}$. A fully Bayesian approach can proceed by sampling from $p(\boldsymbol{\psi}|\mathbf{D})$ in order to average the conditional posterior in (12) with respect to $\boldsymbol{\psi}$.

3 Joint Modeling of Correlated Responses

Building upon the modeling framework of Section 2, we introduce a novel methodology for joint modeling of correlated outputs. We are particularly interested in the case of jointly modeling continuous and binary outputs that are correlated, but our framework is more general. Again denote the process of interest as $\mathbf{f}(\cdot)$, but we now allow its outputs to be of arbitrary form, which could include continuous, binary, categorical, ordinal or other types of output. Let $\mathcal{G}(\cdot)$ be a multivariate function that acts analogous to a link function for general linear models. The function $\mathcal{G}(\cdot)$ is a deterministic function that takes continuous inputs on the real line and maps them to the range of $\mathbf{f}(\cdot)$. We allow for $\mathcal{G}(\cdot)$ to be flexible in its specification and thus only impose the restriction that $\mathcal{G}(\cdot)$ be a function that preserves the range of $\mathbf{f}(\cdot)$. Where the domain and range of $\mathbf{f}(\cdot)$ are identical, then it makes sense for $\mathcal{G}(\cdot)$ to be a one-to-one function. Our framework allows for quite general $\mathcal{G}(\cdot)$, however it is often convenient to decompose $\mathcal{G}$ into $r \leq p$ independent transformations, i.e. $\mathcal{G}^T(\cdot) = (\mathcal{G}_1^T(\cdot), ..., \mathcal{G}_r^T(\cdot))$. Although $\mathcal{G}(\cdot)$ could be any arbitrarily complex multivariate function, we typically use a $\mathcal{G}(\cdot)$ that be decomposed into r independent transformations such that r is equal to the number of unique output types. Thus, each $\mathcal{G}_i(\cdot)$, for $i = 1, .., r$, is a transformation of the inputs to a unique output space. When the outputs do not have the same form for all i we refer to the outputs as mixed type, for example $\mathcal{G}_1$ might be regression outputs and $\mathcal{G}_2$ would be classification outputs. This rule of thumb is suggested in order to facilitate ease in specifying appropriate transformations that preserve the range of $\mathbf{f}(\cdot)$, as well as model tractability in specifying the posterior distributions and the calculations that follow. Following the model formulation in (4), we define $\mathbf{H}$ as before but now let $\mathbf{B} = [\mathbf{B}_1 \cdots \mathbf{B}_r] \in \mathbb{R}^{q \times p}$, where $\mathbf{B}_i \in \mathbb{R}^{q \times p_i}$ is a subset of $\mathbf{B}$, and similarly $\boldsymbol{\omega}(\cdot) = \text{vec}(\boldsymbol{\omega}_1(\cdot), ..., \boldsymbol{\omega}_r(\cdot))$, where $\boldsymbol{\omega}_i(\cdot) \in \mathbb{R}^{p_i}$ is a subset of $\boldsymbol{\omega}(\cdot)$, for $i = 1, ..., r$ and $p = p_1 + ... + p_r$. Now, under our proposed modeling framework, at any collection of inputs $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n) \in \mathcal{X} \subset \mathbb{R}^d$, we model the output with the following general likelihood

$$\mathcal{L}(\mathbf{B}, \mathbf{T}, \boldsymbol{\psi}; \mathbf{Y}) = \prod_{i=1}^{n} p(\mathbf{Y} | \mathcal{G}(\boldsymbol{\mu}(\mathbf{x}_i) + \boldsymbol{\omega}(\mathbf{x}_i))) \tag{15}$$

where we can write the full model as follows:

$$\begin{aligned}
\mathbf{f}(\mathbf{X}) &= \mathcal{G}(\boldsymbol{\mu}(\mathbf{X}) + \boldsymbol{\omega}(\mathbf{X})) \\
\boldsymbol{\mu}(\mathbf{X}) &= (\mathbf{I}_p \otimes \mathbf{H})\,\mathrm{vec}(\mathbf{B}) \\
\boldsymbol{\omega}(\mathbf{X})|\mathbf{T}, \boldsymbol{\phi}, \eta &\sim \mathrm{GP}_p(\mathbf{0}, \mathbf{C}(\mathbf{X}, \mathbf{X}))\,.
\end{aligned} \tag{16}$$

When we can decompose $\mathcal{G}$ into independent transformations, we can rewrite the component-wise model as follows:

$$\begin{aligned}
\mathbf{f}_1(\mathbf{X}) &= \mathcal{G}_1((\mathbf{I}_{p_1} \otimes \mathbf{H})\mathrm{vec}(\mathbf{B}_1) + \boldsymbol{\omega}_1(\mathbf{X})) \\
&\vdots \\
\mathbf{f}_r(\mathbf{X}) &= \mathcal{G}_r((\mathbf{I}_{p_r} \otimes \mathbf{H})\mathrm{vec}(\mathbf{B}_r) + \boldsymbol{\omega}_r(\mathbf{X}))
\end{aligned} \tag{17}$$

where $\mathbf{f}^T(\mathbf{X}) = (\mathbf{f}_1^T(\mathbf{X}), ..., \mathbf{f}_r^T(\mathbf{X}))$ and

$$\boldsymbol{\omega}(\mathbf{X}) = \begin{pmatrix} \boldsymbol{\omega}_1(\mathbf{X}) \\ \vdots \\ \boldsymbol{\omega}_r(\mathbf{X}) \end{pmatrix} \sim \mathrm{GP}_p\left(\mathbf{0}, \mathbf{C}(\mathbf{X}, \mathbf{X})\right). \tag{18}$$

Clearly, under the models in (16) and (17), we have induced a correlation structure between outputs by allowing $\boldsymbol{\omega}(\mathbf{X})$ to be modeled as a joint multivariate Gaussian process. Similar to Albert and Chib (1993), we introduce a latent process structure, $\boldsymbol{\ell}_i = (\mathbf{I}_{p_i} \otimes \mathbf{H})\mathrm{vec}(\mathbf{B}_i) + \boldsymbol{\omega}_i(\mathbf{X})$, by augmenting the model with the unobservable output $\boldsymbol{\ell}_i \in \mathbb{R}^{p_i}$ that allows for a mapping between $\mathbf{f}_i(\mathbf{X})$ and $\boldsymbol{\ell}_i$. Here $\mathcal{G}^T(\boldsymbol{\ell}) = (\mathcal{G}_1^T(\boldsymbol{\ell}_1), ..., \mathcal{G}_r^T(\boldsymbol{\ell}_r))$ is a set of transformations that govern the mapping between the observed outputs and the latent parameters. Formulating our model this way allows for a lot of theoretical carry-over from Section 2 as well as model tractability. Clearly when we allow all of the $\mathcal{G}_i(\boldsymbol{\ell}_i) = \boldsymbol{\ell}_i$ for $i = 1, ..., p$, akin to an identity link, we recover the multivariate Gaussian process model of Section 2.2. Thus, when $\mathcal{G}_i(\boldsymbol{\ell}_i)$ is the identity link, we obtain regression models, and likewise, when $\mathcal{G}_i(\boldsymbol{\ell}_i)$ involves the logistic or probit link, we obtain classification models.

For the applications in this paper, we are interested in jointly modeling a multivariate continuous output and a binary output under our new modeling framework, and so, we

let $\mathbf{f}^T(\mathbf{X}) = (\mathcal{G}_1^T(\boldsymbol{\ell}_1), \mathcal{G}_2^T(\boldsymbol{\ell}_2))$ be a multivariate stochastic process where $\mathcal{G}_1(\boldsymbol{\ell}_1) \in \mathbb{R}^{p-1}$ represents a real continuous output and $\mathcal{G}_2(\boldsymbol{\ell}_2) \in \{0, 1\}$ be a binary output. Now at any collection of inputs $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n) \in \mathcal{X} \subset \mathbb{R}^d$, we model the mixed type output as follows:

$$\mathbf{f}_1(\mathbf{X}) = \mathcal{G}_1(\boldsymbol{\ell}_1) \tag{19}$$

$$\mathbf{f}_2(\mathbf{X}) = \mathcal{G}_2(\boldsymbol{\ell}_2) \tag{20}$$

where

$$\begin{pmatrix} \boldsymbol{\omega}_1(\mathbf{X}) \\ \boldsymbol{\omega}_2(\mathbf{X}) \end{pmatrix} \sim \mathrm{GP}_p\left(\mathbf{0}, \mathbf{C}(\mathbf{X}, \mathbf{X})\right) \tag{21}$$

and $\mathcal{G}_1(\boldsymbol{\ell}_1) = \boldsymbol{\ell}_1$ and

$$\mathcal{G}_2(\ell_{2,i}) = \begin{cases} 1 & \text{if } \ell_{2,i} > 0 \\ 0 & \text{if } \ell_{2,i} \leq 0 \end{cases} \tag{22}$$

where $\ell_{2,i}$ is the i^{th} element of $\boldsymbol{\ell}_2$. We define data and latent parameter matrices $\mathbf{D}$ and $\mathbf{L}$, respectively, such that $\mathrm{vec}(\mathbf{D}) = \mathbf{Y}$ and $\mathrm{vec}(\mathbf{L}) = (\boldsymbol{\ell}_1^T, \boldsymbol{\ell}_2^T)^T$, and paralleling the model in Section 2.2, we use the same joint prior in (6) for $\mathbf{B}$ and $\mathbf{T}$, and arrive at the following joint posterior density of our model

$$\begin{aligned} &p(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \mathbf{B}, \mathbf{T}, \boldsymbol{\psi} | \mathbf{Y}) \propto |\mathbf{R}|^{-p/2} |\mathbf{T}|^{-(n-q+p+1)/2} p(\boldsymbol{\psi}) \\ &\exp\left\{-\frac{1}{2}\left[\mathrm{tr}\left(\mathbf{L}^T\mathbf{G}\mathbf{L}\mathbf{T}^{-1}\right) + \mathrm{tr}\left(\left(\mathbf{B} - \hat{\mathbf{B}}\right)^T \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)\left(\mathbf{B} - \hat{\mathbf{B}}\right)\mathbf{T}^{-1}\right)\right]\right\} \\ &\times \prod_{i=1}^{n(p-1)} \mathbf{1}(\mathbf{Y}_{1,i} = \boldsymbol{\ell}_{1,i}) \prod_{i=1}^{n} \left[\mathbf{1}(\ell_{2,i} > 0)\mathbf{1}(\mathbf{Y}_{2,i} = 1) + \mathbf{1}(\ell_{2,i} \leq 0)\mathbf{1}(\mathbf{Y}_{2,i} = 0)\right] \end{aligned} \tag{23}$$

where $\mathbf{Y}_1$ corresponds to the vector of $n(p-1)$ continuous outputs, $\mathbf{Y}_2$ corresponds to the vector of n binary outputs, $\mathbf{1}(\cdot)$ is an indicator function and $p(\boldsymbol{\psi})$ is an arbitrary proper prior distribution for the correlation parameters in (21). Working with the joint posterior density in (23) is no simple feat, however, inference methods can be devised in order to obtain samples of all of the unknown parameters of the joint posterior density (Section 4.2). Conveniently, integrating out $\mathbf{B}$ and $\mathbf{T}$ from our model is possible and thus we can

derive the joint posterior predictive distribution of $(\tilde{\boldsymbol{\ell}}_1, \tilde{\boldsymbol{\ell}}_2)$, conditional on $\boldsymbol{\ell}_1$, $\boldsymbol{\ell}_2$,$\boldsymbol{\psi}$, and $\mathbf{Y}$ as the following multivariate $\mathcal{T}$ process

$$\begin{pmatrix} \tilde{\boldsymbol{\ell}}_1 \\ \tilde{\boldsymbol{\ell}}_2 \end{pmatrix} \Bigg| \boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\psi}, \mathbf{Y} \sim \mathcal{T}_{mp}\left(\boldsymbol{\mu}^*(\cdot), \hat{\mathbf{T}} \otimes \mathbf{C}^*(\cdot,\cdot), n-q\right) \tag{24}$$

where for $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, ..., \tilde{\mathbf{x}}_m) \in \mathcal{X}$ we have that $\boldsymbol{\mu}^*(\mathbf{X})$ and $\mathbf{C}^*(\mathbf{X}, \mathbf{X})$ are the same as (13) and (14). Obtaining samples of $\tilde{\boldsymbol{\ell}}_1$ and $\tilde{\boldsymbol{\ell}}_2$ from (24) proceeds immediately and allows us to quantify our uncertainty at unobserved inputs for new latent parameters. Likewise, predicting the binary value of $\mathcal{G}_2(\tilde{\boldsymbol{\ell}}_2)$ proceeds by evaluating the following integral (Rasmussen and Williams, 2006)

$$p(\mathbf{f}_2(\tilde{\mathbf{X}}) = 1 | \boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\psi}, \mathbf{Y}) = \int\int p(\mathbf{f}_2(\tilde{\mathbf{X}}) = 1 | \tilde{\boldsymbol{\ell}}_1, \tilde{\boldsymbol{\ell}}_2) p(\tilde{\boldsymbol{\ell}}_1, \tilde{\boldsymbol{\ell}}_2 | \boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\psi}, \mathbf{Y}) d\tilde{\boldsymbol{\ell}}_1 d\tilde{\boldsymbol{\ell}}_2. \tag{25}$$

The integral in (25) is analytically intractable and so we calculate the integral through Monte Carlo integration by first taking many samples of $\tilde{\boldsymbol{\ell}}_1$ and $\tilde{\boldsymbol{\ell}}_2$ from the multivariate $\mathcal{T}$ process in (24) and then passing those samples through the sigmoid function $p(\mathbf{f}_2(\tilde{\mathbf{X}}) = 1 | \tilde{\boldsymbol{\ell}}_1, \tilde{\boldsymbol{\ell}}_2)$, which in our case is a probit function, and then averaging over those values. Classification of the binary outputs can then proceed by classifying outputs as a one if the posterior predictive probability is greater than 0.5 and otherwise classify the output as a zero.

4 Sequential Inference via Particle Learning

4.1 Particle Learning

Particle learning (PL), as established by Carvalho et al. (2010), is a type of sequential Monte Carlo (SMC) algorithm designed for online inference in dynamic models. SMC algorithms are an advantageous alternative to Markov chain Monte Carlo (MCMC) algorithms when the data arrive sequentially in time. MCMC algorithms suffer from a computational burden in online inference, due to the algorithm having to be restarted every time a new data point arrives, whereas in SMC algorithms this is not the case. SMC relies on particles, $\{\mathbf{S}_t^{(i)}\}_{i=1}^N$, containing the sufficient information, $\mathbf{S}_t$, about the uncertainties given data $\mathbf{z}^t = (z_1, ..., z_t)$

up to time t, with N denoting the total number of particles. These particles, at time t, are then used to approximate the posterior distribution, $p(\mathbf{S}_t|\mathbf{z}^t)$, where now online posterior inference continues by updating the particle approximation from time t to time $t+1$. We want to emphasize that the only thing changing in time here is the accumulation of additional data points; our model itself is not dynamic, in that the underlying parameter distributions are the same at all time steps, it is only our posterior estimates that get updated with the arrival of new data points. It can be shown (Gramacy and Polson, 2011) that the PL update to the particles can be derived from the following decomposition

$$\begin{aligned} p(\mathbf{S}_{t+1}|\mathbf{z}^{t+1}) &= \int p(\mathbf{S}_{t+1}|\mathbf{S}_t, z_{t+1}) d\mathbb{P}(\mathbf{S}_t|\mathbf{z}^{t+1}) \\ &\propto \int p(\mathbf{S}_{t+1}|\mathbf{S}_t, z_{t+1}) p(z_{t+1}|\mathbf{S}_t) d\mathbb{P}(\mathbf{S}_t|\mathbf{z}^t). \end{aligned} \tag{26}$$

The above decomposition, (26), suggests that we update the particle approximation in two steps:

1. *Resample*: Sample indices $\zeta(i) \sim \text{Multinomial}(w_t^{(i)}, N)$ where each index is given weight

$$w_t^{(i)} \propto p(z_{t+1}|\mathbf{S}_t^{(i)}) = \int p(z_{t+1}|\mathbf{S}_{t+1}) p(\mathbf{S}_{t+1}|\mathbf{S}_t) d\mathbf{S}_{t+1} \tag{27}$$

   for $i = 1, ..., N$.

2. *Propagate*: Propagate $\mathbf{S}_t^{\zeta(i)}$ to $\mathbf{S}_t^{(i+1)}$ with a draw from $\mathbf{S}_{t+1}^{(i)} \sim p\left(\mathbf{S}_{t+1}|\mathbf{S}_t^{\zeta(i)}, z_{t+1}\right)$ to obtain a new collection of particles $\{\mathbf{S}_{t+1}^{(i)}\}_{i=1}^N \sim p(\mathbf{S}_{t+1}|\mathbf{z}^{t+1})$.

While there exists a plethora of SMC algorithms with components similar to PL, we choose to explore particle learning based methods due to the convenient form of the posterior predictive distribution of Gaussian process models and past successes of particle learning in the univariate Gaussian process setting (Gramacy and Polson, 2011; Liang and Lee, 2014). A slight modification to the above PL steps, similar to the filter of Storvik (2002), would be to reverse the order and first propagate and then resample in similar fashion as above. In the sections that follow, we make use of both the propagate-resample framework as well as the resample-propagate framework although both schemes could be applied in either case. In Section 4.2 we choose to use the propagate-resample scheme of

Storvik (2002) in order to avoid lengthy MCMC steps, which we discuss in Section 4.2, for updating the latent parameters. In Section 4.3 we use the resample-propagate scheme that mimics PL Gaussian process model of Gramacy and Polson (2011).

4.2 Particle Learning Joint Regression and Classification

As a first step, identifying the sufficient information, S_t, is pivotal for any particle learning algorithm. Recalling the joint regression and classification (JRC) model, Section 3, we have that the sufficient information needed for the JRC model is all of the parameters of the correlation matrix in (3), namely, $\boldsymbol{\psi} = \{\boldsymbol{\phi}, \eta\}$. A necessary quantity, we tend to think of the latent parameters $\boldsymbol{\ell}_i$ as also part of the sufficient information. We are able to analytically integrate out $\boldsymbol{\beta}$ and $\mathbf{T}$ from the posterior predictive process in (24) and so we do not consider $\hat{\boldsymbol{\beta}}$ or $\hat{\mathbf{T}}$ as part of the sufficient information since they can be directly calculated from $\boldsymbol{\psi}$. However, for efficient bookkeeping and implementation we do include the correlation matrix from (3) as part of the sufficient information even though it can be directly calculated from $\boldsymbol{\psi}$ as well. Thus, we define the sufficient information as $\mathbf{S}_t = \{\boldsymbol{\psi}_t, \mathbf{R}_t, \mathbf{L}_t\}$ where $\mathbf{L}^t = [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2]$ is the matrix of latent parameters up until time t. Similarly, we define $\mathbf{D}^t$ as the data matrix of all observed outputs up until time t, and $\mathbf{L}_{t+1} = [\tilde{\boldsymbol{\ell}}_1, \tilde{\boldsymbol{\ell}}_2]$ and $\mathbf{D}_{t+1}$ as the matrices for new latent parameters and observed outputs, respectively, at time $t+1$.

Taking a joint improper prior on $(\mathbf{B}, \mathbf{T})$ requires initializing the particles conditional on $t_0 > q + p$ data points to ensure a proper posterior. Recalling that we can integrate $\mathbf{B}$ and $\mathbf{T}$ out of (23), we obtain our initial particles using a Metropolis-Hasting scheme to sample $\boldsymbol{\psi}_{t_0}^{(i)}$, $\boldsymbol{\ell}_{1,t_0}^{(i)}$, and $\boldsymbol{\ell}_{2,t_0}^{(i)}$ from $p(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\psi}|\mathbf{D}^{t_0})$. Once sampled, we then deterministically calculate $\mathbf{R}_{t_0}^{(i)}$ from $\boldsymbol{\psi}_{t_0}^{(i)}$ and thereby obtain $\mathbf{S}_{t_0}^{(i)}$. After initialization of the particles, we can follow the *propagate* and *resample* steps for updating particles $\{\mathbf{S}_t^{(i)}\}_{i=1}^N$ to $\{\mathbf{S}_{t+1}^{(i)}\}_{i=1}^N$. We outline the three steps as the following:

1. *Propagate*: The first propagate step draws new latent parameters $\{\mathbf{L}_{t+1}^{(i)}\}_{i=1}^N$ from $p(\mathbf{L}_{t+1}|\mathbf{S}_t^{(i)})$ via sampling from the posterior predictive distribution in (24).

2. *Resample*: The resample step requires sampling indices $\zeta^{(i)} \sim \text{Multinomial}(w_t^{(i)}, N)$

where each index is given weight

$$w_t^{(i)} \propto p(\mathbf{D}_{t+1}|\mathbf{L}_{t+1}^{(i)}, \mathbf{S}_t^{(i)}), \tag{28}$$

for $i = 1, ..., N$, which can be factored into a product of a multivariate normal distribution and a Bernoulli distribution.

3. *Propagate*: The second propagate step updates each resampled sufficient information $\mathbf{S}_t^{\zeta^{(i)}}$ to $\mathbf{S}_{t+1}^{(i)}$ by accounting for the new data output $\mathbf{D}_{t+1}$. Since the correlation parameters of the JRC model are not dynamic, we may propagate deterministically each resampled $\boldsymbol{\psi}$ by copying $\boldsymbol{\psi}_t^{\zeta^{(i)}}$ to $\boldsymbol{\psi}_{t+1}^{(i)}$. A key step in deterministically propagating components of $\mathbf{S}_t^{\zeta^{(i)}}$ to $\mathbf{S}_{t+1}^{(i)}$ requires the calculation of the propagated correlation function $\mathbf{R}_{t+1}$. Once we have deterministically propagated $\boldsymbol{\psi}_t^{\zeta^{(i)}}$ to $\boldsymbol{\psi}_{t+1}^{(i)}$ we calculate the propagated correlation matrix as follows

$$\mathbf{R}_{t+1}^{(i)} = \begin{bmatrix} \mathbf{R}_t^{(i)} & (\tilde{\mathbf{F}}^{(i)})^T \\ \tilde{\mathbf{F}}^{(i)} & \tilde{\mathbf{R}}^{(i)} \end{bmatrix} \tag{29}$$

where $\tilde{\mathbf{F}} = \rho(\mathbf{x}_{t+1}, \mathbf{X})$ and $\tilde{\mathbf{R}} = \rho(\mathbf{x}_{t+1}, \mathbf{x}_{t+1})$.

As noted in Section 4.1, we could instead have implemented a resample-propagate scheme, similar to Gramacy and Polson (2011) for GP classification models, but this would then make our propagate step dependent upon sampling the new latent parameters from $p(\mathbf{L}_{t+1}|\mathbf{S}_t, \mathbf{D}_{t+1})$ which would need to be done in an MCMC scheme that would require a large number of Metropolis-Hastings updates to ensure stationarity, and this would be done within each particle, and thus considerably slowing the PL algorithm. Conversely, following a propagate-resample scheme requires no Metropolis-Hastings updates and hence no concerns over convergence within each propagate step of the latent parameters.

Like many sequential Monte Carlo algorithms, particle learning is susceptible to particle degeneracy in future resample steps. Particle degeneracy occurs when, after some iterations of the algorithm, all but one particle will have weights that are very close to zero. In particular, deterministically propagating components of $\mathbf{S}_t^{\zeta^{(i)}}$ to $\mathbf{S}_{t+1}^{(i)}$ almost surely guarantees particle degeneracy in the future resampling steps. However, deterministic propagation does lead to significant speed gains in computation. In order to take advantage of the

computational speed up of deterministic propagation, and to avoid degeneracy, we include a *rejuvenate* step inside of the *propagate* step that samples from the posterior distribution in (8) via MCMC for the sake of rejuvenating the particles (MacEachern et al., 1999). Following the lead of Gramacy and Polson (2011) we find a single Metropolis-Hastings step for the parameters of $\boldsymbol{\psi}_t$, for each particle, to be sufficient in avoiding particle degeneracy.

4.3 Particle Learning Multivariate Gaussian Process

Recall the fact that when we allow $\mathcal{G}_i(\boldsymbol{\ell}_i) = \boldsymbol{\ell}_i$, akin to an identity link, we recover the multivariate Gaussian process model of Section 2.2. As a special case, utilizing particle learning for inference in this model parallels the particle learning joint regression and classification (PLJRC) model but in far more simplicity. Given the separable multivariate Gaussian process, Section 2.2, the sufficient information needed for this model is the same as the sufficient information of the JRC model, i.e., $\mathbf{S}_t = \{\boldsymbol{\psi}_t, \mathbf{R}_t, \mathbf{L}^t\}$. However, given the form of $\mathcal{G}(\boldsymbol{\ell})$, we no longer need to worry about $\mathbf{L}^t$ as part of the sufficient information, since we now effectively treat the latent parameters as the true observed data, and so we set $\mathbf{S}_t = \{\boldsymbol{\psi}_t, \mathbf{R}_t\}$ and $\mathbf{L}^t = \mathbf{D}^t$.

Similarly, taking a joint improper prior on $(\mathbf{B}, \mathbf{T})$ requires us to initialize our particles conditional on $t_0 > q + p$ data points to ensure a proper posterior. We follow the same Metropolis-Hasting scheme described in Section 4.2. After initialization of the particles, we can follow the two step *resample* and *propagate* steps for updating particles $\{\mathbf{S}_t^{(i)}\}_{i=1}^N$ to $\{\mathbf{S}_{t+1}^{(i)}\}_{i=1}^N$. We argue, as Gramacy and Polson (2011) do, that the multivariate Gaussian process is not a dynamic model, i.e. its parameters do not change in time, and so our resample step only requires us to consider $\mathbf{L}_{t+1}$ conditional on $\mathbf{S}_{t+1}$ rather than the typical integration over $p(\mathbf{S}_{t+1}|\mathbf{S}_t)$. We outline the two steps as the following:

1. *Resample*: The resample step requires sampling indices $\zeta(i) \sim \text{Multinomial}(w_t^{(i)}, N)$ where each index is given weight

$$w_t^{(i)} \propto p(\mathbf{D}_{t+1}|\mathbf{S}_t^{(i)}) = \frac{p(\mathbf{D}_{t+1}|\mathbf{S}_t^{(i)})}{\sum_{i=1}^N p(\mathbf{D}_{t+1}|\mathbf{S}_t^{(i)})} \tag{30}$$

for $i = 1, ..., N$. Conveniently, $p(\mathbf{D}_{t+1}|\mathbf{S}_t^{(i)})$ is just the probability of $\mathbf{D}_{t+1}$ under the multivariate $\mathcal{T}$ process (13–14) given $\mathbf{S}_t^{(i)}$.

2. *Propagate*: The multivariate Gaussian process is not a dynamic model we may propagate deterministically each resampled sufficient information by copying $\mathbf{S}_t^{\zeta(i)}$ to $\mathbf{S}_{t+1}^{(i)}$. Like before, Section 4.2, once we have deterministically propagated $\boldsymbol{\psi}_t^{\zeta(i)}$ to $\boldsymbol{\psi}_{t+1}^{(i)}$ we calculate the propagated correlation matrix following (29).

5 Illustrating Examples

As a proof of concept, we demonstrate the applicability of the models presented in Sections 4.2 and 4.3 with a number of illustrative examples and comparisons with previous work. Because our motivating example is a computer modeling problem, we follow the standard approach in the computer modeling literature (Santner et al., 2003) by using a Gaussian correlation function with unknown length-scale parameter ϕ and nugget η for all of these examples, i.e.,

$$\rho(\mathbf{x}_j, \mathbf{x}_k; \boldsymbol{\phi}, \eta) = \exp\left(-\sum_{i=1}^{d} \frac{|x_{ij} - x_{ik}|^2}{\phi_i}\right) + \eta\delta_{j,k} \tag{31}$$

where $\delta_{\cdot,\cdot}$ is the Kronecker delta function. Our methodology applies to any valid choice of correlation function, but here we follow the substantial literature in computer modeling. For the length-scale parameter ϕ, we use the prior suggested in Gramacy and Lee (2008) and let $\phi \sim \frac{1}{2}(\text{Gamma}(1, 20) + \text{Gamma}(10, 10))$. The prior for ϕ encodes our belief that about half of the particles should represent Gaussian process parameterizations with wavy surfaces while the other half should represent Gaussian process parameterizations which are quite smooth or approximately linear. We place a prior on the nugget parameter, $\eta \sim \text{Exp}(10)$, that allows for a moderate amount of noise, or provides robustness in fitting (Gramacy and Lee, 2012). Lastly, we specify our matrix regression $\mathbf{H}$, with regression functions $\mathbf{h}(\mathbf{x}_1), ..., \mathbf{h}(\mathbf{x}_n)$, such that it is equivalent to a linear regression model with an intercept term.

5.1 PLMGP Synthetic Data Example

To illustrate the particle learning multivariate Gaussian process (PLMGP) method, examples based on simulated data are presented. We first consider the one-dimensional synthetic sinusoidal data used originally by Higdon (2002) and later by Gramacy and Polson (2011). Now, in the case of the one-dimensional sinusoidal functions we have that

$$f(x) = \sin\left(\frac{\pi x}{5}\right) + \frac{1}{5}\cos\left(\frac{4\pi x}{5}\right) \quad \text{and} \quad g(x) = \sin\left(\frac{f(x)}{3}\right) \tag{32}$$

where $x \in [0, 10]$. We simulate data, $\mathbf{y}(x) = (z(x), g(x))$, from (32) with noise such that $z(x) \sim N(f(x), \sigma = 0.1)$. We start with an initial sample of five data points from the model (32) that are uniformly spaced throughout the domain of x, see Figure 1, and sequentially sample thirty more data points from a Uniform(0,10) distribution. Using our PLMGP algorithm, with $N = 2000$ particles, we obtain posterior predictive surfaces for $f(x)$ and $g(x)$ that capture the true data generating models, without noise, very well (Figure 1).

Alternatively, we could ignore the correlation between the outputs of $f(x)$ and $g(x)$ and use the particle learning Gaussian processes (PLGP) (Gramacy and Polson, 2011) to model $f(x)$ and $g(x)$ independently. As an additional experiment, we randomly generated 450 data sets from the data generating model in (32). Each of the 450 data sets started with an initial sample of five data points and sequentially added thirty more data points as seen previously. For all 450 data sets we ran both the PLMGP and PLGP algorithms, with $N = 2000$ particles, and compared the performance of the algorithms based on two separate metrics: mean squared error and coverage. When running the PLGP algorithm we defaulted to using the `R plgp` package library (Gramacy, 2012).

We calculate the distribution of the mean square errors of the fits under both the PLMGP and PLGP models (Figure 2). On average, the fits under the PLMGP framework do a better job and lead to smaller mean squared error values. Likewise, for the 450 data sets we calculated the coverage as the percentage of time that the true output, $f(x)$ and $g(x)$, was covered (pointwise) by the 95% credible intervals. A summary of the numerical results are found in Figure 2. When modeling the data from $f(x)$, the PLMGP method does slightly better at being centered around the nominal 95% coverage rate while, on average, the PLGP method seemed seemed to undercover. On the other hand, when modeling

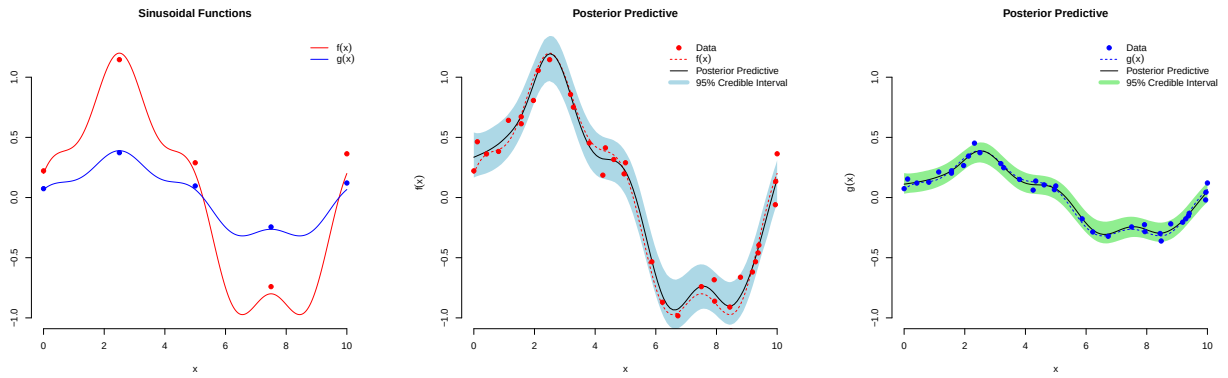

Figure 1: The true data generating models $f(x)$ and $g(x)$ with five observed data points (left). Also, posterior predictive surfaces of $f(x)$ (middle) and $g(x)$ (right) with 95% credible intervals.

$g(x)$ both methods did a good job at being centered around the nominal 95% coverage rate. We attribute these facts to higher amount of nonstationarity needed to model $f(x)$ and to the lower amount of nonstationarity for $g(x)$. Sharing information by modeling data from $f(x)$ and $g(x)$ jointly is an advantage of the PLMGP that appears to help when modeling correlated functions that require differing amounts of nonstationary modeling. However, in both instances, the minimum observed coverage rate was lower for the PLMGP than the PLGP.

Given our two metrics, the PLMGP method appears to outperform the PLGP method when modeling functions that are clearly correlated. This fact should come as no surprise since the PLMGP is able to take advantage of the shared information that comes from correlated processes whereas the independent PLGP cannot. It is worth noting though that although the PLMGP outperformed the PLGP, the PLGP method still did very well at modeling the true functions.

As a further comparison, we contrasted fitting the model using MCMC versus the proposed PL scheme. Working under the same scenario as before, we generated 35 data points from (32), but treated the design as static and fixed, and so we started both the MCMC and PLMGP method with 5 data points and sequentially added 30 more data points to each. This was repeated 50 times in an `R` implementation on an Intel Core-i7

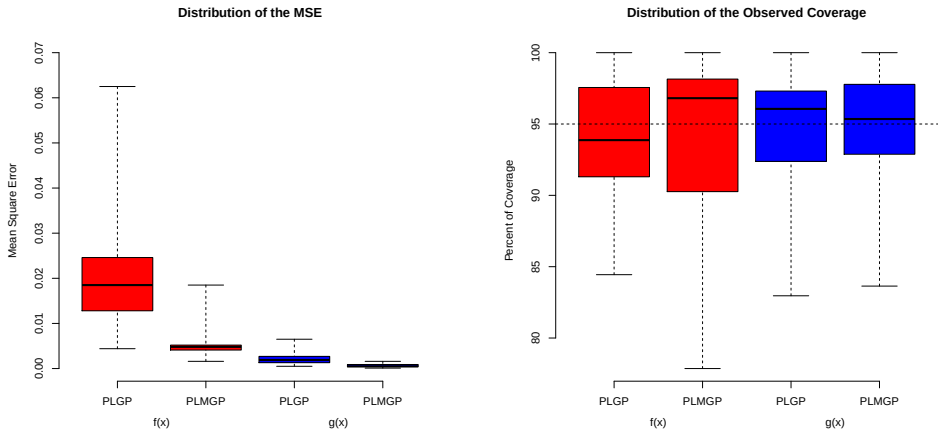

Figure 2: Summary of the mean square errors (left) and of the observed coverage (right) under both models for 450 repeated designs. The nominal coverage was taken to be 95% (horizontal dashed line).

2.9GHz CPU computer. The efficiency of the two approaches were judged based on the average computing time and average mean square error over the 50 repetitions. The average mean square error (and the standard deviation of the mean square error in parentheses) for predicting $f(x)$ was 0.0056 (0.0015) for PLMGP, and 0.0062 (0.0026) for MCMC. The average (sd) mean square error for predicting $g(x)$ was 0.0006 (0.0001) for PLMGP, and 0.0007 (0.0001) for MCMC. Only slightly better, there did not seem to be a large difference in the prediction error between fitting the model with MCMC versus PLMGP. There was however a stark contrast in average computing time where it took, on average, 25 minutes for one repetition based on 100,000 MCMC iterations and only 6 minutes, on average, for the PLMGP method to finish one repetition with $N = 4000$ particles. Clearly, the computational gain in speed is what sets the PLMGP method apart from the traditional MCMC methods when data arrives sequentially.

Although many sequential Monte Carlo methods have to account for the problem of particle depletion, in our application of PLMGP, we had effective sample sizes which were always above the common threshold of $N/2$ (Prado and West, 2010).

5.2 PLJRC Synthetic Data Example

Now, consider data generated from the following one-dimensional dampened cosine functions, $f(x)$, used by Santner et al. (2003), $g(x)$, and the step function $h(x)$

$$f(x) = \exp(-1.4x)\cos\left(\frac{7\pi x}{2}\right), \quad g(x) = \exp(-3x)\cos\left(\frac{7\pi x}{2}\right)$$
$$\text{and} \quad h(x) = \begin{cases} 1 & \text{if } f(x) + g(x) > 0 \\ 0 & \text{if } f(x) + g(x) \leq 0 \end{cases} \tag{33}$$

where $x \in [0, 1]$. Clearly, $f(x)$ and $g(x)$ are continuous functions for all x while $h(x)$ is a binary function that is highly correlated with $f(x)$ and $g(x)$. We start with an initial sample of ten data points from the functions in (33), where we sample our inputs $x \in [0, 1]$ from a Latin hypercube design (McKay et al. (1979); Urban and Fricker (2010)) and sequentially sample ten more inputs in [0,1] from the same Latin hypercube design. Using the particle learning joint regression and classification (PLJRC) algorithm, with $N = 4000$ particles, we initialize our particles and sequentially update our model using the remaining ten data points. When finished, we obtain posterior predictive surfaces for $f(x)$ and $g(x)$ that do an excellent job of capturing the true underlying data generating functions (Figure 3). Moreover, we are able to calculate the posterior predicted probability of $h(x) = 1$ by evaluating the expression in (25) via Monte Carlo integration. We classify $h(x)$ as 1 if the posterior predicted probability is greater than 0.5 and classify $h(x)$ as 0 otherwise. In this example (Figure 3), this method leads to a very low misclassification rate of 4%.

In order to compare the performance of including correlation between outputs, we can ignore the correlation structure between $f(x)$, $g(x)$ and $h(x)$ and use the PLGP method of Gramacy and Polson (2011) and model the posterior predictive probability associated with $h(x) = 1$ independently of $f(x)$ and $g(x)$. As a test, we randomly generated 450 data sets from the data generating functions in (33) and ran both the PLGP and PLJRC algorithms in order to compare the mean squared error and classifications rates under each model. When running the PLGP algorithm we again defaulted to using the `R plgp` package library. Each of the 450 data sets started with an initial sample of ten data points and sequentially added ten more. The mean square error was always better for the PLJRC method than the PLGP (Figure 4), and the classification rate was overall better, suggesting

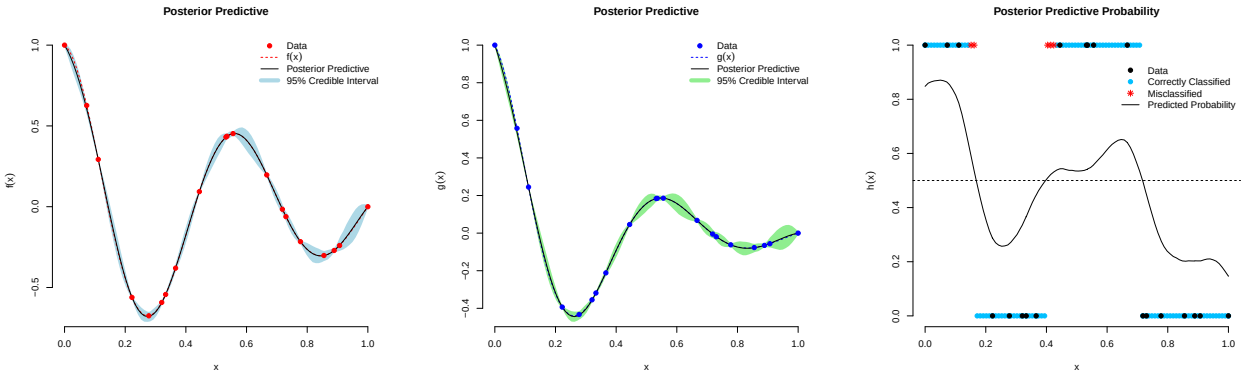

Figure 3: The posterior predictive surface and 95% credible intervals for $f(x)$ (left), posterior predictive surface and 95% credible intervals for $g(x)$ (middle), and posterior predictive probability associated with $h(x)$ (right). When the posterior predicted probability exceeds 0.5 we predict $h(x) = 1$ and $h(x) = 0$ otherwise.

the strength of modeling $f(x)$, $g(x)$, and $h(x)$ jointly when correlated.

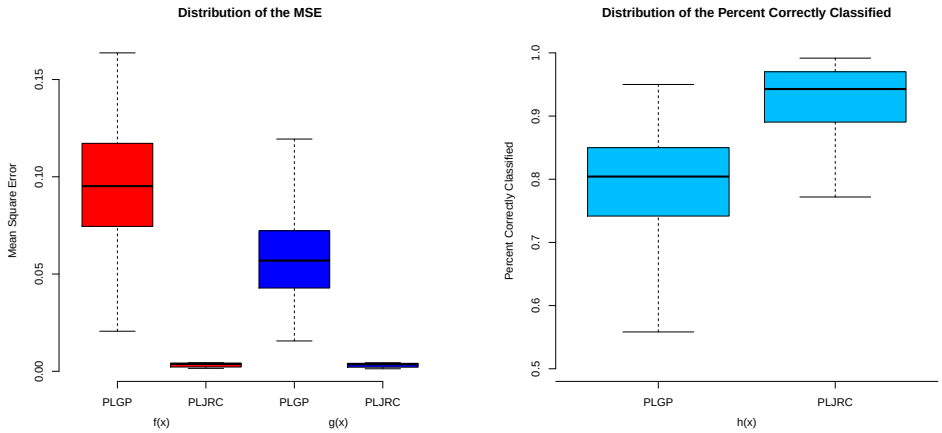

Figure 4: Summary of the mean squared error (left) and classification rates (right) under both models for 450 repeated designs for data simulated using (33).

As a last comparison, we investigated the difference in fitting the model using MCMC versus the proposed PLJRC scheme. Working under the same scenario as before, we generated 20 data points from (33), but treated the design as static and fixed, and so we started both the MCMC and PLJRC method with 10 data points and sequentially added 10 more data points to each. This was repeated 50 times in an `R` implementation on an Intel Core-i7

2.9GHz CPU computer. The efficiency of the two approaches were judged based on the average computing time, average mean square error, and average correct classification rate over the 50 repetitions. The average (sd) mean square error for predicting $f(x)$ was 0.0038 (0.0023) for PLJRC, and 0.0044 (0.0019) for MCMC. The average (sd) mean square error for predicting $g(x)$ was 0.0034 (0.0021) for PLMGP, and 0.0036 (0.0026) for MCMC. As before, we only found slightly better differences in prediction error between fitting the model with MCMC versus PLJRC. On the other hand, classification was slightly better using MCMC rather than PLJRC with a correct classification rate of 2.8% versus 4.1%, respectively. However, the difference in average computing time between the two methods is large enough to preclude the use of MCMC over PLJRC. On average, it took 22 minutes for one repetition based on 100,000 MCMC iterations and only 4 minutes, on average, for the PLMGP method to finish one repetition with $N = 4000$ particles. The computational burden of MCMC is a clear disadvantage in sequential inference as compared to the much quicker sequential Monte Carlo.

Comparable to Section 5.1, in our application of PLJRC, we had effective sample sizes which were always above the common threshold of $N/2$ (Prado and West, 2010).

5.3 Hydraulic Capture Problem

A real world application, the hydraulic capture problem from the community problems (Mayer et al., 2002) involves a groundwater contamination scenario based on the U.S. Geological Survey computer simulator MODFLOW (McDonald and Harbaugh, 1996). The objective of the problem is to contain a plume of contaminated groundwater by installing up to four wells to control the direction of groundwater movement, and to do so at minimal cost. The MODFLOW simulator was built to model this physical process where the inputs to the computer simulator are the coordinates, (x_1, x_2), of the well and its pumping rate, x_3. We focus, as Lindberg and Lee (2015) do, on the single well version of the problem and further constrain ourselves to the same initial conditions as Lindberg and Lee (2015). Thus we focus our search of the input space on the region with $235 \leq x_1 \leq 270$, $580 \leq x_2 \leq 680$, and $-0.0064 \leq x_3 \leq -0.0051$, where negative rates indicate extraction. Lindberg and Lee (2015) focused their efforts on searching this restricted area of the input space due to the

fact that only about 2% of initial inputs, based on a Latin hypercube design (LHD), will yield a valid output in the original input space. Narrowing the search to a smaller region of the input space increases the number of valid outputs to about 5%.

We reformulate this problem in the framework of a constrained optimization problem as

$$\min_{\mathbf{x}}\{f(\mathbf{x}) : g(\mathbf{x}) = 1;\ 235 \leq x_1 \leq 270; 580 \leq x_2 \leq 680; -0.0064 \leq x_3 \leq -0.0051\} \quad (34)$$

where the objective f we wish to minimize describes the cost required to install and operate the wells. The contaminant plume is contained when the binary constraint, g, is met. The objective f and constraint g are both highly complex and nonlinear functions in which outputs from each function can only be obtained from running the computer simulator. However, worst of all, when the constraint g is not met, the computer simulator only tells us that the constraint was not met but gives us no information about the output of the objective function f. The time it takes to run the computer simulator is nontrivial and so it not feasible to run the computer simulator at every possible combination of inputs and find the one that optimizes the problem (34). Instead, we proceed by constructing a surrogate model (Sacks et al., 1989; Santner et al., 2003) sequentially while searching for the minimum of the response surface (Jones et al., 1998; Taddy et al., 2009). We construct our surrogate model by modeling both the continuous objective f and the binary constraint g jointly using our PLJRC model. Modeling f and g as correlated makes sense in this context, because extracting more fluid is more likely to meet the constraint (contain the plume of contamination) but will cost more; extracting less fluid will be cheaper but less likely to meet the constraint. The model is sequentially updated by selecting new candidate inputs, $\mathbf{x}_*$, that maximize the probability of finding a smaller objective function value f. Our approach toward this goal is that of expected improvement (Jones et al., 1998). We define the improvement statistic at a proposed input $\mathbf{x}$ to be

$$I(\mathbf{x}) = \max\{f_{\min} - f(\mathbf{x}), 0\} \quad (35)$$

where, after N runs of the simulator, $f_{\min} = \min\{f(\mathbf{x}_1), ..., f(\mathbf{x}_N)\}$ is the current minimum value observed. Since the proposed input $\mathbf{x}$ has not yet been observed, $f(\mathbf{x})$ is unknown and, conditional on the observed inputs $\mathbf{x}_1, ..., \mathbf{x}_N$, the distribution of $f(\mathbf{x})$ can be represented

by using the posterior predictive distribution of the PLJRC. Now that $I(\mathbf{x})$ can be regarded as a random variable, we choose new candidate inputs, $\mathbf{x}_*$, by selecting those inputs that maximize the expected improvement

$$\mathbf{x}_* = \arg\max_{\mathbf{x}\in\mathcal{X}} \mathbb{E}\{I(\mathbf{x})\}. \tag{36}$$

Fortunately, conditional on a particular parameterization of the PLJRC, the expected improvement is available in closed form as

$$\mathbb{E}\{I(\mathbf{x})\} = (f_{\min} - \mu(\mathbf{x}))\Phi\left(\frac{f_{\min} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) + \sigma(\mathbf{x})\phi\left(\frac{f_{\min} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) \tag{37}$$

where $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are the mean and standard deviation of the posterior predictive distribution of $f(\mathbf{x})$ and $\Phi(\cdot)$ and $\phi(\cdot)$ are the standard normal cdf and pdf respectively. The equation in (37) provides a combined measure of how promising a candidate point is, that trades off between local search ($\mu(\mathbf{x})$ under $f_{\min}$) and global search ($\sigma(\mathbf{x})$). However, in the presence of constraints, it makes no sense to search for candidate inputs that maximize the expected improvement yet violate the constraints. Thus, we go one step further and choose candidate inputs, $\mathbf{x}_*$, by selecting those inputs that maximize the following (Lindberg and Lee, 2015)

$$\mathbf{x}_* = \arg\max_{\mathbf{x}\in\mathcal{X}} \mathbb{E}\{I(\mathbf{x})\}^{\alpha_1} S(\mathbf{x})^{\alpha_2} \tag{38}$$

where $S(\mathbf{x})$ is the asymmetric entropy defined as

$$S(\mathbf{x}) = \frac{2p(\mathbf{x})(1-p(\mathbf{x}))}{p(\mathbf{x}) - 2wp(\mathbf{x}) + w^2} \tag{39}$$

and $p(\mathbf{x})$ is the predicted probability that the constraint is satisfied at the input $\mathbf{x}$. Here, α_1, α_2 and w are constants that we set equal to the default values suggested in Lindberg and Lee (2015), and so, $\alpha_1 = 1, \alpha_2 = 5$ and $w = 2/3$. Thus, maximizing the quantity in (38) is a trade off between maximizing the expected improvement and the probability of meeting the constraints. Calculating the probability that an input $\mathbf{x}_*$ satisfies the constraint, i.e. $p(\mathbf{x}) = \Pr(g(\mathbf{x}) = 1)$, is not a trivial problem, however, we make use of the posterior predictive probability calculations for $g(\mathbf{x})$ under our PLJRC model in order to make calculations of the probability of meeting the constraint. In many constrained minimization problems, the probability of the constraint is correlated with the response

function, so that the expected improvement is in opposition to the probability of meeting the constraint, which is what makes the problem difficult. Our JRC model can handle this correlation, in contrast to the existing models in the computer experiment literature.

To show the advantage of modeling the objective and constraint as correlated, we followed the same setup as Lindberg and Lee (2015) and began with an initial sample size of 65 inputs, based on a LHD, to run the MODFLOW simulator at. The output from the MODFLOW simulator was then used to fit our PLJRC model where we chose to use $N = 2000$ particles. The search for the optimal solution then proceeded by sequentially selecting 300 more inputs to run the MODFLOW simulator at based on choosing points that maximized the expected constrained improvement in (38). We follow the strategy of Taddy et al. (2009) and select the candidate set of inputs from a LHD of size 500 times the input dimension augmented by an additional 10% of the candidate locations taken from a smaller LHD bounded to within 5% of the domain range of the current best point. This hybrid space filling design further ensures that our algorithm searches both locally as well as globally. The best solution our algorithm found is a cost of \$22,952.8 by setting the coordinates of the pump to $(x_1, x_2) = (258.8451, 638.3419)$ and the extraction rate to $x_3 = -0.005320973$. Our solution reached the same cost values found in Lindberg and Lee (2015) and Lee et al. (2011). Although under slightly different initial conditions, our optimal cost found was much better than all eight of the solutions found in Fowler et al. (2008), with the best solution reported there being \$23,421 found using Implicit Filtering for Constrained Optimization (IFFCO). Likewise, we reran the same analysis 30 times, and the average value found after 300 runs reached by our algorithm (\$22,953.3), Figure 5, was much lower than the best values found by eight competing optimization algorithms as reported in Fowler et al. (2008).

We ran the independent PLGP methodology 30 times under the same setup and conditions as the joint PLJRC methodology and found that both methods lead to comparable optimal solutions. Over those 30 Monte Carlo repetitions, the independent PLGP models also found an optimal cost of \$22,952.8. However, it was clear that independent PLGP

model did not always find an optimal solution within the 300 sequential updates, see Figure 5, whereas the PLJRC model consistently found the optimal solution in fewer than 150 sequential updates. Thus, it would seem that the PLJRC model was able to take advantage of the correlation structure between the objective and constraint functions that the independent PLGP model was not.

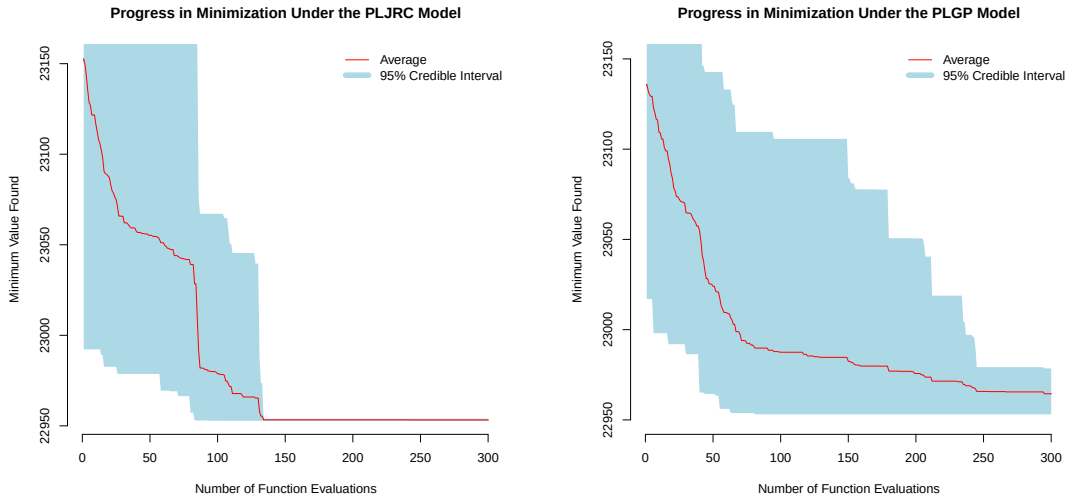

Figure 5: After 300 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint (left) and independent (right) models. The plot shows the average best minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best minumums found.

The framework of expected constrained improvement is heavily reliant on the model behind both the objectives and constraints. Here we demonstrate that our PLJRC model does a very good job at modeling the objective and constraint surfaces and allows the constrained expected improvement mechanism to outperform other established solutions. In constrained optimization, the problems are typically difficult because the objective function and the constraint are in opposition, i.e., strongly negatively correlated. Our model performs better by fully modeling this correlation.

6 Discussion

We have established a novel methodological framework for modeling mixed type outputs with fully Bayesian inference. By introducing latent outputs we are able to jointly model correlated outputs. In particular, we showed how our framework could be applied in the case of jointly modeling continuous and binary outputs that were correlated, particularly for constrained optimization of computer simulation experiments. We combined the strengths of our joint regression and classification (JRC) model with the speed of sequential Monte Carlo methods to conduct fast inference. Sequential inference for the JRC model based on MCMC is inefficient because it requires rerunning the MCMC every time a new data point arrives. A considerable reduction in computation time is achieved by applying particle learning methodology to the JRC model. Thus, the marriage of the JRC model with particle learning (PLJRC) is indeed a powerful new technique. We highlighted the strengths of the PLJRC over other competing methodologies (such as the PLGP) with simulated examples and were able to show its practical usefulness in a real-world constrained optimization problem of a computer simulator.

The introduction of latent parameters for joint modeling underlies the innovation of our model. By directly allowing the latent parameters to be the observed outputs, in an identity link fashion, we can recover the multivariate Gaussian process model of Section 2. Furthermore, this also allows us to extend the particle learning algorithm to the model of Section 2, thereby creating a new stochastic modeling technique (PLMGP) comparable to that of Gramacy and Polson (2011). This framework also provides a novel implementation of particle learning for multivariate Gaussian processes.

An obvious extension within our framework would be to extend the model to a richer class of outputs. Using suitable transformations $\mathcal{G}(\cdot)$, we can envision an infinite continuum of correlated data types that could be modeled jointly.

Acknowledgments

We thank the associate editor and two anonymous reviewers for their detailed and insightful comments that have helped improve this paper.

References

Albert, J. H. and Chib, S. (1993), "Bayesian Analysis of Binary and Polychotomous Response Data," *Journal of the American Statistical Association*, 88, 669–679.

Banerjee, S. and Gelfand, A. E. (2002), "Prediction, Interpolation, and Regression for Spatial Misaligned Data Points," *Sankhya*, 64, 227–245.

Banerjee, S., Gelfand, A. E., and Carlin, B. P. (2004), *Hierarchical Modeling and Analysis for Spatial Data*, Chapman & Hall/CRC, New York.

Bonilla, E. V., Chai, K. M., and Williams, C. K. I. (2008), "Multi-task Gaussian Process Prediction," in *Advances in Neural Information Processing Systems 20*, eds. J. C. Platt, D. Koller, Y. Singer, and S. Rowesi, Cambridge, MA, MIT Press.

Carvalho, C. M., Johannes, M. S., Lopes, H. F., and Polson, N. G. (2010), "Particle Learning and Smoothing," *Statistical Science*, 25, 88–106.

Chan, A. B. (2013), "Multivariate Generlaized Gaussian Process Models," Tech. rep., Department of Computer Science, City University of Hong Kong.

Conti, S. and O'Hagan, A. (2010), "Bayesian emulation of complex multi-output and dynamic computer models," *Journal of Statistical Planning and Inference*, 140, 640–651.

Cressie, N. A. C. (1993), *Statistics for Spatial Data, revised edition*, John Wiley & Sons.

Diggle, P. J. and Ribeiro Jr., P. J. (2007), *Model-based Geostatistics*, Springer.

Fowler, K. R., Reese, J. P., Kees, C. E., Dennis Jr., J. E., Kelley, C. T., Miller, C. T., Audet, C., Booker, A. J., Couture, G., Darwin, R. W., Farthing, M. W., Finkel, D. E., Gablonsky, G., Gray, G., , and Kolda, T. G. (2008), "A Comparison of Derivative-free

Optimization Methods for Water Supply and Hydraulic Capture Community Problem," *Advances in Water Resources*, 31, 743–757.

Fricker, T. E., Oakley, J. E., and Urban, N. M. (2013), "Multivariate Gaussian Process Emulators with Nonseperable Covariance Stuctures," *Technometrics*, 55, 47–56.

Gaspari, G. and Cohn, S. E. (1999), "Construction of correlation functions in two and three dimensions," *Quarterly Journal of the Royal Metereological Society*, 125, 723–757.

Gelfand, A. E., Schmidt, A. M., Banerjee, S., and Sirmans, C. F. (2004), "Nonstationary Multivariate Process Modeling through Spatially Varying Coregionalization," *Test*, 13, 263–312.

Goulard, M. and Voltz, M. (1992), "Linear coregionalization model: Tools for estimation and choice of cross-covariogram matrix," *Mathematical Geology*, 21, 269–286.

Gramacy, R. B. (2012), *plgp: Particle Learning of Gaussian Processes*, R package version 1.1-5.

Gramacy, R. B. and 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.

Gramacy, R. B. and Lee, H. K. H. (2012), "Cases for the Nugget in Modeling Computer Experiments," *Statistics and Computing*, 22, 713–722.

Gramacy, R. B. and Polson, N. G. (2011), "Particle learning of Gaussian process models for sequential design and optimization," *Journal of Computational and Graphical Statistics*, 20, 102–118.

Gupta, A. K. and Nagar, D. K. (2000), *Matrix Variate Distributions*, Chapman & Hall/CRC Press, Boca Raton, Florida.

Hayashi, K., Takenouchi, T., Tomioka, R., and Kashima, H. (2012), "Self-measuring Similarity for Multi-task Gaussian Process." in *ICML Unsupervised and Transfer Learning*, eds. I. Guyon, G. Dror, V. Lemaire, G. W. Taylor, and D. L. Silver, vol. 27 of *JMLR Proceedings*, pp. 145–154, JMLR.org.

Higdon, D. (2002), "Space and Space–time Modeling Using Process Convolutions," in *Quantitative Methods for Current Environmental Issues*, eds. C. Anderson, V. Barnett, P. C. Chatwin, and A. H. El-Shaarawi, pp. 37–56, London, Springer-Verlag.

Jones, D., Schonlau, M., and Welch, W. J. (1998), "Efficient Global Optimization of Expensive Black Box Functions," *Journal of Global Optimization*, 13, 455–492.

Lee, H. K. H., Gramacy, R. B., Linkletter, C., and Gray, G. A. (2011), "Optimization Subject to Hidden Constraints via Statistical Emulation," *Pacific Journal of Optimization*, 7, 467–478.

Liang, W. W. J. and Lee, H. K. H. (2014), "Sequential Process Convolution Gaussian Process Models via Particle Learning," *Statistics and Its Interface*, 7, 465–475.

Lindberg, D. and Lee, H. K. H. (2015), "Optimization Under Constraints by Applying an Asymmetric Entropy Measure," *Journal of Computational and Graphical Statistics*, 24, 379–393.

Liu, X., Chen, F., Lu, Y., and Lu, C. (2013), "Spatial Prediction of Large Multivariate Non-Gaussian Data," Tech. rep., Department of Computer Science, Virginia Tech.

MacEachern, S. M., Clyde, M., and Liu, J. S. (1999), "Sequential Importance Sampling for Nonparametric Bayes Models: The Next Generation," *Canadian Journal of Statistics*, 27, 251–267.

Majumdar, A. and Gelfand, A. E. (2007), "Multivariate Spatial Modeling for Geostatistical Data Using Convolved Covariance Functions," *Mathematical Geology*, 39, 229–245.

Mardia, K. V. and Goodall, C. R. (1993), "Spatial–Temporal Analysis of Multivariate Environmental Monitoring Data," *Multivariate Environmental Statistics*, 6, 76.

Mayer, A. S., Kelley, C. T., and Miller, C. T. (2002), "Optimal Design for Problems Involving Flow and Transport Phenomena in Saturated Subsurface Systems," *Advances in Water Resources*, 25.

McDonald, M. G. and Harbaugh, A. W. (1996), "Programmers documentation for MODFLOW-96, an update to the U. S. geological survey modular finite difference ground-water flow model," Tech. rep., Open-File Report 96-486, U. S. Geological Survey.

McKay, M. D., Conover, W. J., and Beckman, R. J. (1979), "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, 21, 239–245.

Montagna, S. and Tokdar, S. T. (2013), "Computer emulation with non-stationary Gaussian processes," Tech. rep., Duke University.

Moustaki, I. and Knott, M. (2000), "Generalized Latent Trait Models," *Pyschometrika*, 65, 391–411.

Neal, R. M. (1999), "Regression and classification using Gaussian process priors (with discussion)," in *Bayesian Statistics 6*, ed. e. a. J. M. Bernardo, pp. 476–501, Oxford University Press.

Prado, R. and West, M. (2010), *Time Series: Modelling, Computation & Inference*, Chapman & Hall/CRC Press, Boca Raton, FL.

Rasmussen, C. E. and Williams, C. K. I. (2006), *Gaussian Processes for Machine Learning*, The MIT Press, Cambridge, MA.

Rowe, D. B. (2003), *Multivariate Bayesian Statitics: Models for Source Seperation and Signal Unmixing*, Chapman & Hall/CRC, Boca Raton, FL.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989), "Design and Analysis of Computer Experiments," *Statistical Science*, 4, 409–435.

Sammel, M. D., Ryan, L. M., and Legler, J. M. (1997), "Latent Variable Models for Mixed Discrete and Continuous Outcomes," *Journal of the Royal Statistical Society. Series B (Methodological)*, 59, 667–678.

Santner, T. J., Williams, B. J., and Notz, W. I. (2003), *The Design and Analysis of Computer Experiments*, Springer-Verlag, New York, NY.

Storvik, G. (2002), "Particle Filters in State Space Models with the Presence of Unknown Static Parameters," *IEEE TRans. Signal Process*, 50, 281–289.

Taddy, M., Lee, H. K. H., Gray, G. A., and Griffin, J. D. (2009), "Bayesian Guided Pattern Search for Robust Local Optimization," *Technometrics*, 51, 389–401.

Urban, N. M. and Fricker, T. E. (2010), "A Comparison of Latin Hypercube and Grid Ensemble Designs for the Multivariate Emulation of an Earth System Model," *Computers and Geosciences*, 36, 746–755.

Wackernagel, H. (2003), *Multivariate Geostatistics: An Introduction with Applications*, Springer, New York, NY.

Xu, Z., Yan, F., and Qi, Y. (2012), "Infinite Tucker Decomposition: Nonparametric Bayesian Models for Multiway Data Analysis," in *Proceedings of the 29th International Conference on Machine Learning*, ICML '12.

Yu, K., Tresp, V., and Schwaighofer, A. (2005), "Learning Gaussian Processes from Multiple Tasks," in *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pp. 1012–1019, New York, NY, USA, ACM.

Zhe, S., Qi, Y., Youngja, P., Molloy, I., and Chari, S. (2013), "Dintucker: Scaling Up Gaussian Process Models on Multidimensional Arrays with Billions of Elements," Tech. rep., arXiv preprint arXiv:1311.2663.

Zhe, S., Xu, Z., Chu, X., Qi, Y., and Park, Y. (2015), "Scalable Nonparametric Multiway Data Analysis," *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pp. 1125–1134.