
Article:
Alvarez, M.A., Rosasco, L. and Lawrence, N.D. (2012) Kernels for Vector-Valued Functions: a Review. Foundations and Trends® in Machine Learning, 4 (3). pp. 195-266. ISSN: 1935-8237

https://doi.org/10.1561/2200000036

Kernels for Vector-Valued Functions: a Review

Mauricio A. Álvarez[+], Lorenzo Rosasco[♯,†], Neil D. Lawrence[⋆,◇],

‡ - *School of Computer Science, University of Manchester Manchester, UK, M13 9PL.*

\+ *Department of Electrical Engineering, Universidad Tecnológica de Pereira, Colombia, 660003*

♯ - *CBCL, McGovern Institute, Massachusetts Institute of Technology, Cambridge, MA, USA*

† -*IIT@MIT Lab, Istituto Italiano di Tecnologia, Genova, Italy*

⋆ - *Department of Computer Science, University of Sheffield, UK*

◇ *The Sheffield Institute for Translational Neuroscience, Sheffield, UK.*

malvarez@utp.edu.co, lrosasco@mit.edu, n.lawrence@sheffield.ac.uk

April 17, 2012

Abstract

Kernel methods are among the most popular techniques in machine learning. From a regularization perspective they play a central role in regularization theory as they provide a natural choice for the hypotheses space and the regularization functional through the notion of reproducing kernel Hilbert spaces. From a probabilistic perspective they are the key in the context of Gaussian processes, where the kernel function is known as the covariance function. Traditionally, kernel methods have been used in supervised learning problem with scalar outputs and indeed there has been a considerable amount of work devoted to designing and learning kernels. More recently there has been an increasing interest in methods that deal with multiple outputs, motivated partly by frameworks like multitask learning. In this paper, we review different methods to design or learn valid kernel functions for multiple outputs, paying particular attention to the connection between probabilistic and functional methods.

Contents

1 Introduction

Many modern applications of machine learning require solving several decision making or prediction problems and exploiting dependencies between the problems is often the key to obtain better results and coping with a lack of data (to solve a problem we can *borrow strength* from a distinct but related problem).

In *sensor networks*, for example, missing signals from certain sensors may be predicted by exploiting their correlation with observed signals acquired from other sensors [72]. In *geostatistics*, predicting the concentration of heavy pollutant metals, which are expensive to measure, can be done using inexpensive and oversampled variables as a proxy [37]. In *computer graphics*, a common theme is the animation and simulation of physically plausible humanoid motion. Given a set of poses that delineate a particular movement (for example, walking), we are faced with the task of completing a sequence by filling in the missing frames with natural-looking poses. Human movement exhibits a high-degree of correlation. Consider, for example, the way we walk. When moving the right leg forward, we unconsciously prepare the left leg, which is currently touching the ground, to start moving as soon as the right leg reaches the floor. At the same time, our hands move synchronously with our legs. We can exploit these implicit correlations for predicting new poses and for generating new natural-looking walking sequences [106]. In *text categorization*, one document can be assigned to multiple topics or have multiple labels [50]. In all the examples above, the simplest approach ignores the potential correlation among the different output components of the problem and employ models that make predictions individually for each output. However, these examples suggest a different approach through a joint prediction exploiting the interaction between the different components to improve on individual predictions. Within the machine learning community this type of modeling is often broadly referred to to as *multitask learning*. Again the key idea is that information shared between different tasks can lead to improved performance in comparison to learning the same tasks individually. These ideas are related to *transfer learning* [97, 20, 12, 74], a term which refers to systems that learn by transferring knowledge between different domains, for example: "what can we learn about running through seeing walking?"

More formally, the classical supervised learning problem requires estimating the output for any given input $\mathbf{x}_*$; an estimator $f_*(\mathbf{x}_*)$ is built on the basis of a training set consisting of N input-output pairs $S = (\mathbf{X}, \mathbf{Y}) = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$. The input space $\mathcal{X}$ is usually a space of vectors, while the output space is a space of *scalars*. In multiple output learning (MOL) the output space is a space of *vectors*; the estimator is now a *vector valued function* $\mathbf{f}$. Indeed, this situation can also be described as the problem of solving D distinct classical supervised problems, where each problem is described by one of the components $f_1, \ldots, f_D$ of $\mathbf{f}$. As mentioned before, the key idea is to work under the assumption that the problems are in some way related. The idea is then to exploit the relation among the problems to improve upon solving each problem separately.

The goal of this survey is twofold. First, we aim at discussing recent results in multi-output/multi-task learning based on kernel methods and Gaussian processes providing an account of the state of the art in the field. Second, we analyze systematically the connections between Bayesian and regularization (frequentist) approaches. Indeed, related techniques have been proposed from different perspectives and drawing clearer connections can boost advances in the field, while fostering collaborations between different communities.

The plan of the paper follows. In chapter 2 we give a brief review of the main ideas underlying kernel methods for scalar learning, introducing the concepts of regularization in reproducing kernel Hilbert spaces and Gaussian processes. In chapter 3 we describe how similar concepts extend to the context of vector valued functions and discuss different settings that can be considered. In chapters 4 and 5 we discuss approaches to constructing multiple output kernels, drawing connections between the Bayesian and regularization frameworks. The parameter estimation problem and the computational complexity problem are both described in chapter 6. In chapter 7 we discuss some potential applications that can be seen as multi-output learning. Finally we conclude in chapter 8 with some remarks and discussion.

2 Learning Scalar Outputs with Kernel Methods

To make the paper self contained, we will start our study reviewing the classical problem of learning a scalar valued function, see for example [100, 40, 10, 82]. This will also serve as an opportunity to review connections between Bayesian and regularization methods.

As we mentioned above, in the classical setting of supervised learning, we have to build an estimator (e.g. a classification rule or a regression function) on the basis of a training set $S = (\mathbf{X}, \mathbf{Y}) = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$. Given a symmetric and positive bivariate function $k(\cdot, \cdot)$, namely a *kernel*, one of the most popular estimators in machine learning is defined as

$$f_*(\mathbf{x}_*) = \mathbf{k}_{\mathbf{x}_*}^\top (k(\mathbf{X}, \mathbf{X}) + \lambda N \mathbf{I})^{-1} \mathbf{Y}, \tag{1}$$

where $k(\mathbf{X}, \mathbf{X})$ has entries $k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{Y} = [y_1, \ldots, y_N]^\top$ and $\mathbf{k}_{\mathbf{x}_*} = [k(\mathbf{x}_1, \mathbf{x}_*), \ldots, k(\mathbf{x}_N, \mathbf{x}_*)]^\top$, where $\mathbf{x}_*$ is a new input point. Interestingly, such an estimator can be derived from two different, though, related perspectives.

2.1 A Regularization Perspective

We will first describe a regularization (frequentist) perspective (see [35, 105, 100, 86]). The key point in this setting is that the function of interest is assumed to belong to a reproducing kernel Hilbert space (RKHS),

$$f_* \in \mathcal{H}_k.$$

Then the estimator is derived as the minimizer of a regularized functional

$$\frac{1}{N}\sum_{i=1}^{N}(f(\mathbf{x}_i) - y_i)^2 + \lambda\|f\|_k^2. \tag{2}$$

The first term in the functional is the so called empirical risk and it is the sum of the squared errors. It is a measure of the price we pay when predicting $f(\mathbf{x})$ in place of y. The second term in the functional is the (squared) norm in a RKHS. This latter concept plays a key role, so we review a few essential concepts (see [87, 6, 105, 25]). A RKHS $\mathcal{H}_k$ is a Hilbert space of functions and can be defined by a reproducing kernel[1] $k : \mathcal{X} \times \mathcal{X} \to \mathbf{R}$, which is a symmetric, positive definite function. The latter assumption amounts to requiring the matrix with entries $k(\mathbf{x}_i, \mathbf{x}_j)$ to be positive for any (finite) sequence $(\mathbf{x}_i)$. Given a kernel k, the RKHS $\mathcal{H}_k$ is the Hilbert space such that the function $k(\mathbf{x}, \cdot)$ belongs to belongs to $\mathcal{H}_k$ for all $\mathbf{x} \in \mathcal{X}$ and

$$f(\mathbf{x}) = \langle f, k(\mathbf{x}, \cdot)\rangle_k, \quad \forall\, f \in \mathcal{H}_k,$$

where $\langle \cdot, \cdot \rangle_k$ is the inner product in $\mathcal{H}_k$.

The latter property, known as the reproducing property, gives the name to the space. Two further properties make RKHS appealing:

- functions in a RKHS are in the closure of the linear combinations of the kernel at given points, $f(\mathbf{x}) = \sum_i k(\mathbf{x}_i, \mathbf{x})c_i$. This allows us to describe, in a unified framework, linear models as well as a variety of generalized linear models;
- the norm in a RKHS can be written as $\sum_{i,j} k(\mathbf{x}_i, \mathbf{x}_j)c_i c_j$ and is a natural measure of how *complex* is a function. Specific examples are given by the shrinkage point of view taken in ridge regression with linear models [40] or the regularity expressed in terms of magnitude of derivatives, as is done in spline models [105].

In this setting the functional (2) can be derived either from a regularization point of view [35, 105] or from the theory of empirical risk minimization (ERM) [100]. In the former, one observes that, if the space $\mathcal{H}_k$ is large enough, the minimization of the empirical error is ill-posed, and in particular it responds in an unstable manner to noise, or when the number of samples is low Adding the squared norm stabilizes the problem. The latter point of view, starts from the analysis of ERM showing that generalization to new samples can be achieved if there is a tradeoff between fitting and complexity[2] of the estimator. The functional (2) can be seen as an instance of such a trade-off.

The explicit form of the estimator is derived in two steps. First, one can show that the minimizer of (2) can always be written as a linear combination of the kernels centered at the training set points,

$$f_*(\mathbf{x}_*) = \sum_{i=1}^{N} k(\mathbf{x}_*, \mathbf{x}_i)c_i = \mathbf{k}_{\mathbf{x}_*}^\top \mathbf{c},$$

see for example [65, 19]. The above result is the well known representer theorem originally proved in [51] (see also [88] and [26] for recent results and further references). The explicit form of the coefficients $\mathbf{c} = [c_1, \ldots, c_N]^\top$ can be then derived by substituting for $f_*(\mathbf{x}_*)$ in (2).

[1] In the following we will simply write kernel rather than reproducing kernel.

[2] For example, a measure of complexity is the *Vapnik–Chervonenkis dimension* [86]

2.2 A Bayesian Perspective

A Gaussian process (GP) is a stochastic process with the important characteristic that any finite number of random variables, taken from a realization of the GP, follows a joint Gaussian distribution. A GP is usually used as a prior distribution for functions [82]. If the function f follows a Gaussian process we write

$$f \sim \mathcal{GP}(m, k),$$

where m is the mean function and k the covariance or kernel function. The mean function and the covariance function completely specify the Gaussian process. In other words the above assumption means that for any finite set $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ if we let $f(\mathbf{X}) = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^\top$ then

$$f(\mathbf{X}) \sim \mathcal{N}(m(\mathbf{X}), k(\mathbf{X}, \mathbf{X})),$$

where $m(\mathbf{X}) = [m(\mathbf{x}_1), \ldots, m(\mathbf{x}_N)]^\top$ and $k(\mathbf{X}, \mathbf{X})$ is the kernel matrix. In the following, unless otherwise stated, we assume that the mean vector is zero.

From a *Bayesian* point of view, the Gaussian process specifies our prior beliefs about the properties of the function we are modeling. Our beliefs are updated in the presence of data by means of a *likelihood function*, that relates our prior assumptions to the actual observations. This leads to an updated distribution, the *posterior distribution*, that can be used, for example, for predicting test cases.

In a regression context, the likelihood function is usually Gaussian and expresses a linear relation between the observations and a given model for the data that is corrupted with a zero mean Gaussian noise,

$$p(y|f, \mathbf{x}, \sigma^2) = \mathcal{N}(f(\mathbf{x}), \sigma^2),$$

where σ^2 corresponds to the variance of the noise. Noise is assumed to be independent and identically distributed. In this way, the likelihood function factorizes over data points, given the set of inputs $\mathbf{X}$ and σ^2. The posterior distribution can be computed analytically. For a test input vector $\mathbf{x}_*$, given the training data $\mathbf{S} = \{\mathbf{X}, \mathbf{Y}\}$, this posterior distribution is given by,

$$p(f(\mathbf{x}_*)|\mathbf{S}, \mathbf{x}_*, \boldsymbol{\phi}) = \mathcal{N}(f_*(\mathbf{x}_*), k_*(\mathbf{x}_*, \mathbf{x}_*)),$$

where $\boldsymbol{\phi}$ denotes the set of parameters which include the variance of the noise, σ^2, and any parameters from the covariance function $k(\mathbf{x}, \mathbf{x}')$. Here we have

$$\begin{aligned} f_*(\mathbf{x}_*) &= \mathbf{k}_{\mathbf{x}_*}^\top (k(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I})^{-1} \mathbf{Y}, \\ k_*(\mathbf{x}_*, \mathbf{x}_*) &= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_{\mathbf{x}_*}^\top (k(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_{\mathbf{x}_*} \end{aligned}$$

and finally we note that if we are interested into the distribution of the noisy predictions, $p(y(\mathbf{x}_*)|\mathbf{S}, \mathbf{x}_*, \boldsymbol{\phi})$, it is easy to see that we simply have to add σ^2 to the expression for the predictive variance (see [82]).

Figure 1 represents a posterior predictive distribution for a data vector $\mathbf{Y}$ with $N = 4$. Data points are represented as dots in the figure. The solid line represents the mean function predicted, $f_*(\mathbf{x}_*)$, while the shaded region corresponds to two standard deviations away from the mean. This shaded region is specified using the predicted covariance function, $k_*(\mathbf{x}_*, \mathbf{x}_*)$. Notice how the uncertainty in the prediction increases as we move away from the data points.

Equations for $f_*(\mathbf{x}_*)$ and $k_*(\mathbf{x}_*, \mathbf{x}_*)$ are obtained under the assumption of a Gaussian likelihood, common in regression setups. For non-Gaussian likelihoods, for example in classification problems, closed form solutions are not longer possible. In this case, one can resort to different approximations, including the Laplace approximation and variational methods [82].

2.3 A Connection Between Bayesian and Regularization Point of Views

Connections between regularization theory and Gaussian process prediction or Bayesian models for prediction have been pointed out elsewhere [78, 105, 82]. Here we just give a very brief sketch of the argument. We restrict ourselves to finite dimensional RKHS. Under this assumption one can show that every RKHS can be described in terms of a feature map [100], that is a map $\Phi : \mathcal{X} \to \mathbf{R}^p$, such that

$$k(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{p} \Phi^j(\mathbf{x}) \Phi^j(\mathbf{x}').$$

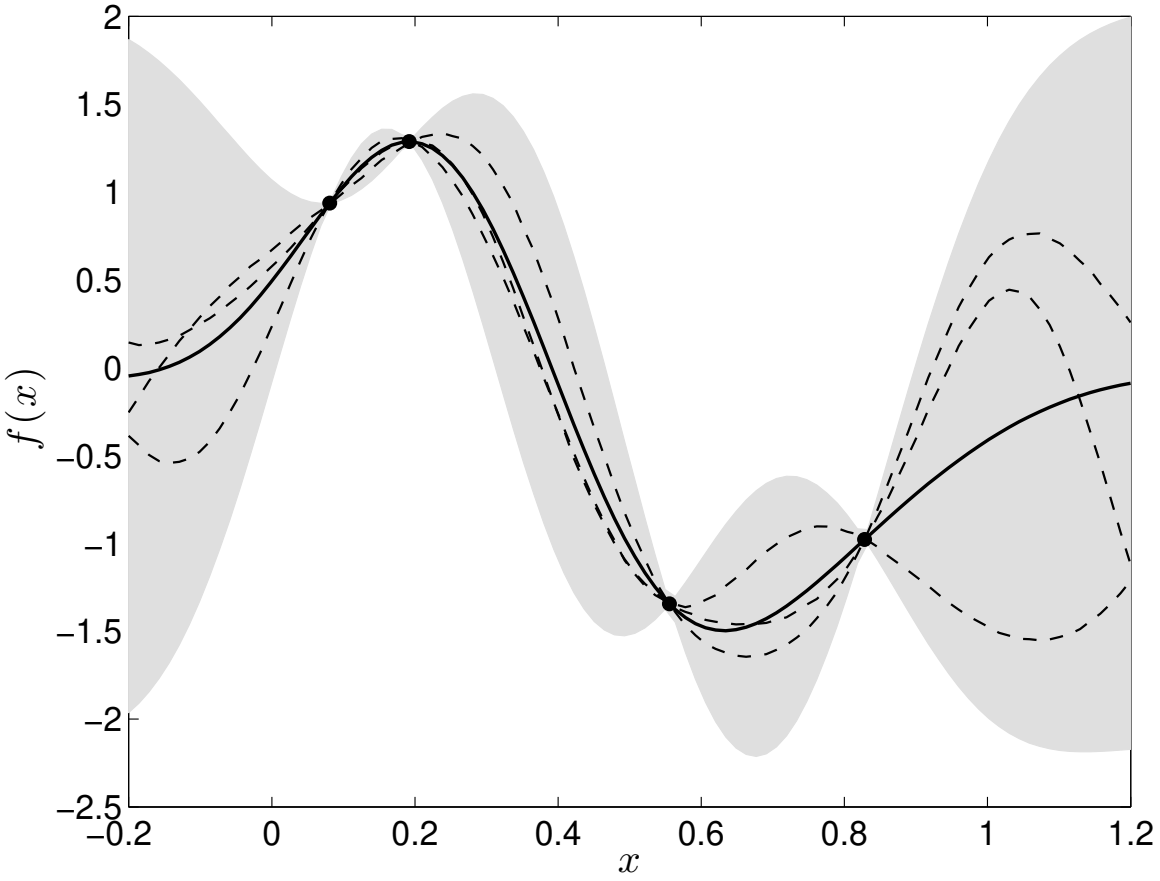

Figure 1: Example of a predictive posterior distribution inferred with $N = 4$. The solid line corresponds to the predictive mean, the shaded region corresponds to two standard deviations of the prediction. Dots are values of the output function $\mathbf{Y}$. We have also included some samples from the posterior distribution, shown as dashed lines.

In fact in this case one can show that functions in the RKHS with kernel k can be written as

$$f_w(\mathbf{x}) = \sum_{j=1}^{p} \mathbf{w}^j \Phi^j(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle, \quad \text{and} \quad \|f_{\mathbf{w}}\|_k = \|\mathbf{w}\|.$$

Then we can build a Gaussian process by assuming the coefficient $w = w^1, \ldots, w^p$ to be distributed according to a multivariate Gaussian distribution. Roughly speaking, in this case the assumption $f_* \sim \mathcal{GP}(0, k)$ becomes

$$\mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_p) \propto e^{-\|\mathbf{w}\|^2}.$$

As we noted before if we assume a Gaussian likelihood we have

$$P(\mathbf{Y}|\mathbf{X}, f) = \mathcal{N}(f(\mathbf{X}), \sigma^2 \mathbf{I}_D) \propto e^{-\frac{1}{\sigma^2}\|f_w(\mathbf{X}) - \mathbf{Y}\|_n^2},$$

where $f_{\mathbf{w}}(\mathbf{X}) = (\langle \mathbf{w}, \Phi(\mathbf{x}_1) \rangle, \ldots, \langle \mathbf{w}, \Phi(\mathbf{x}_n) \rangle)$ and $\|f_w(\mathbf{X}) - \mathbf{Y}\|_n^2 = \sum_{i=1}^{n} (\langle w, \Phi(\mathbf{x}_i) \rangle - y_i)^2$. Then the posterior distribution is proportional to

$$e^{-(\frac{1}{\sigma^2}\|f_{\mathbf{w}}(\mathbf{X}) - \mathbf{Y}\|_n^2 + \|\mathbf{w}\|^2)},$$

and we see that a maximum a posteriori estimate will in turn give the minimization problem defining Tikhonov regularization [98], where the regularization parameter is now related to the noise variance.

We note that in regularization the squared error is often replaced by a more general error term $\frac{1}{N}\sum_{i=1}^{N} \ell(f(\mathbf{x}_i), y_i)$. In a regularization perspective, the *loss function* $\ell : \mathbf{R} \times \mathbf{R} \to \mathbf{R}^+$ measure the error we incur when predicting $f(\mathbf{x})$ in place of y. The choice of the loss function is *problem dependent*. Often used examples are the square loss, the logistic loss or the hinge loss used in support vector machines (see [86]).

The choice of a loss function in a regularization setting can be contrasted to the choice of the likelihood in a Bayesian setting. In this context, the likelihood function models how the observations deviate from the assumed *true* model in the generative process. The notion of a loss function is philosophically different. It represents the cost we pay for making errors. In Bayesian modeling decision making is separated from inference. In the inference stage the posterior distributions are computed evaluating the uncertainty in the model. The loss function appears only at the second stage of the analysis, known as the *decision* stage, and weighs how incorrect decisions are penalized given the current uncertainty. However, whilst the two notions are philosophically very different, we can see that, due to the formulation of the frameworks, the loss function and the log likelihood provide the same role mathematically.

The discussion in the previous sections shows that the notion of a kernel plays a crucial role in statistical modeling both in the Bayesian perspective (as the covariance function of a GP) and the regularization perspective (as a reproducing kernel). Indeed, for scalar valued problems there is a rich literature on the design of kernels (see for example [86, 90, 82] and references therein). In the next sections we show how the concept of a kernel can be used in multi-output learning problems. Before doing that, we describe how the concepts of RKHSs and GPs translate to the setting of vector valued learning.

3 Learning Multiple Outputs with Kernels Methods

In this chapter we discuss the basic setting for learning vector valued functions and related problems (multiclass, multilabel) and then describe how the concept of kernels (reproducing kernels and covariance function for GP) translate to this setting.

3.1 Multi-output Learning

The problem we are interested in is that of learning an unknown functional relationship f between an input space $\mathcal{X}$, for example $\mathcal{X} = \mathbf{R}^p$, and an output space $\mathbf{R}^D$. In the following we will see that the problem can be tackled either assuming that $\mathbf{f}$ belongs to reproducing kernel Hilbert space of vector valued functions or assuming that $\mathbf{f}$ is drawn from a vector valued Gaussian process. Before doing this we describe several related settings all falling under the framework of multi-output learning.

The natural extension of the traditional (scalar) supervised learning problem is the one we discussed in the introduction, when the data are pairs $S = (\mathbf{X}, \mathbf{Y}) = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$. For example this is the typical setting for problems such as motion/velocity fields estimation. A special case is that of multi-category classification problem or multi-label problems, where if we have D classes each input point can be associated to a (binary) coding vector where, for example 1 stands for presence (0 for absence) of a class instance.The simplest example is the so called *one vs all* approach to multiclass classification which, if we have $\{1, \ldots, D\}$ classes, amounts to the coding $i \to \mathbf{e}_i$, where $(\mathbf{e}_i)$ is the canonical basis of $\mathbf{R}^D$.

A more general situation is that where different outputs might have different training set cardinalities, different input points or in the extreme case even different input spaces. More formally, in this case we have a training set $S_d = (\mathbf{X}_d, \mathbf{Y}_d) = (\mathbf{x}_{d,1}, y_{d,1}), \ldots, (\mathbf{x}_{d,N_d}, y_{d,N_d})$ for each component f_d, with $d = 1, \ldots, D$, where the number of data associated with each output, (N_d) might be different and the input for a component might belong to different input space $(\mathcal{X}_d)$.

The terminology used in machine learning often does not distinguish the different settings above and the term multitask learning is often used. In this paper we use the term multi-output learning or vector valued learning to define the general class of problems and use the term multi-task for the case where each component has different inputs. Indeed in this very general situation each component can be thought of as a distinct task possibly related to other tasks (components). In the geostatistics literature, if each output has the same set of inputs the model is called *isotopic* and *heterotopic* if each output to be associated with a different set of inputs [104]. Heterotopic data is further classified into *entirely heterotopic data*, where the variables have no sample locations in common, and *partially heterotopic data*, where the variables share some sample locations. In machine learning, the partially heterotopic case is sometimes referred to as *asymmetric multitask learning* [112, 21].

The notation in the multitask learning scenario (heterotopic case) is a bit more involved. To simplify the notation we assume that the number of data for each output is the same. Moreover, for the sake of simplicity sometimes

we restrict the presentation to the isotopic setting, though the models can usually readily be extended to the more general setting. We will use the notation $\mathbf{X}$ to indicate the collection of all the training input points, $\{\mathbf{X}_j\}_{j=1}^N$, and $\mathbf{S}$ to denote the collection of all the training data. Also we will use the notation $\mathbf{f}(\mathbf{X})$ to indicate a vector valued function evaluated at different training points. This notation has slightly different meaning depending on the way the input points are sampled. If the input to all the components are the same then $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N$ and $\mathbf{f}(\mathbf{X}) = f_1(\mathbf{x}_1), \ldots, f_D(\mathbf{x}_N)$. If the input for the different components are different then $\mathbf{X} = \{\mathbf{X}_d\}_{d=1}^D = \mathbf{X}_1, \ldots, \mathbf{X}_D$, where $\mathbf{X}_d = \{\mathbf{x}_{d,n}\}_{n=1}^N$ and $\mathbf{f}(\mathbf{X}) = (f_1(\mathbf{x}_{1,1}), \ldots, f_1(\mathbf{x}_{1,N})), \ldots, (f_D(\mathbf{x}_{D,1}), \ldots, f_D(\mathbf{x}_{D,N}))$.

3.2 Reproducing Kernel for Vector Valued Function

The definition of RKHS for vector valued functions parallels the one in the scalar, with the main difference that the reproducing kernel is now *matrix* valued, see for example [65, 19] . A reproducing kernel is a symmetric function $\mathbf{K} : \mathcal{X} \times \mathcal{X} \to \mathbf{R}^{D \times D}$, such that for any $\mathbf{x}, \mathbf{x}'$ $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ is a positive semi-definite *matrix*. A vector valued RKHS is a Hilbert space $\mathcal{H}$ of functions $\mathbf{f} : \mathcal{X} \to R^D$, such that for very $\mathbf{c} \in \mathbf{R}^D$, and $\mathbf{x} \in \mathcal{X}$, $\mathbf{K}(\mathbf{x}, \mathbf{x}')\mathbf{c}$, as a function of $\mathbf{x}'$ belongs to $\mathcal{H}$ and moreover $\mathbf{K}$ has the reproducing property

$$\langle \mathbf{f}, \mathbf{K}(\cdot, \mathbf{x})\mathbf{c} \rangle_{\mathbf{K}} = \mathbf{f}(\mathbf{x})^\top \mathbf{c},$$

where $\langle \cdot, \cdot \rangle_{\mathbf{K}}$ is the inner product in $\mathcal{H}$.

Again, the choice of the kernel corresponds to the choice of the representation (parameterization) for the function of interest. In fact any function in the RKHS is in the closure of the set of linear combinations

$$\mathbf{f}(\mathbf{x}) = \sum_{i=1}^{p} \mathbf{K}(\mathbf{x}_i, \mathbf{x})\mathbf{c}_j, \quad \mathbf{c}_j \in \mathbf{R}^D,$$

where we note that in the above equation each term $\mathbf{K}(\mathbf{x}_i, \mathbf{x})$ is a matrix acting on a vector $\mathbf{c}_j$. The norm in the RKHS typically provides a measure of the complexity of a function and this will be the subject of the next sections.

Note that the definition of vector valued RKHS can be described in a component-wise fashion in the following sense. The kernel $\mathbf{K}$ can be described by a scalar kernel R acting jointly on input examples and task indices, that is

$$(\mathbf{K}(\mathbf{x}, \mathbf{x}'))_{d,d'} = R((\mathbf{x}, d), (\mathbf{x}', d')), \tag{3}$$

where R is a scalar reproducing kernel on the space $\mathcal{X} \times \{1, \ldots, D\}$. This latter point of view is useful while dealing with multitask learning, see [28] for a discussion.

Provided with the above concepts we can follow a regularization approach to define an estimator by minimizing the regularized empirical error (2), which in this case can be written as

$$\sum_{j=1}^{D} \frac{1}{N} \sum_{i=1}^{N} (f_j(\mathbf{x}_i) - y_{j,i})^2 + \lambda \|\mathbf{f}\|_{\mathbf{K}}^2, \tag{4}$$

where $\mathbf{f} = (f_1, \ldots, f_D)$. Once again the solution is given by the representer theorem [65]

$$\mathbf{f}(\mathbf{x}) = \sum_{i=1}^{N} \mathbf{K}(\mathbf{x}_i, \mathbf{x})\mathbf{c}_i,$$

and the coefficient satisfies the linear system

$$\overline{\mathbf{c}} = (\mathbf{K}(\mathbf{X}, \mathbf{X}) + \lambda N \mathbf{I})^{-1} \overline{\mathbf{y}}, \tag{5}$$

where $\overline{\mathbf{c}}, \overline{\mathbf{y}}$ are ND vectors obtained concatenating the coefficients and the output vectors, and $\mathbf{K}(\mathbf{X}, \mathbf{X})$ is an $ND \times ND$ with entries $(\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j))_{d,d'}$, for $i, j = 1, \ldots, N$ and $d, d' = 1, \ldots, D$ (see for example [65]). More explicitly

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \begin{bmatrix} (\mathbf{K}(\mathbf{X}_1, \mathbf{X}_1))_{1,1} & \cdots & (\mathbf{K}(\mathbf{X}_1, \mathbf{X}_D))_{1,D} \\ (\mathbf{K}(\mathbf{X}_2, \mathbf{X}_1))_{2,1} & \cdots & (\mathbf{K}(\mathbf{X}_2, \mathbf{X}_D))_{2,D} \\ \vdots & \cdots & \vdots \\ (\mathbf{K}(\mathbf{X}_D, \mathbf{X}_1))_{D,1} & \cdots & (\mathbf{K}(\mathbf{X}_D, \mathbf{X}_D))_{D,D} \end{bmatrix} \tag{6}$$

where each block $(\mathbf{K}(\mathbf{X}_i, \mathbf{X}_j))_{i,j}$ is an N by N matrix (here we make the simplifying assumption that each output has same number of training data). Note that given a new point $\mathbf{x}_*$ the corresponding prediction is given by

$$\mathbf{f}(\mathbf{x}_*) = \mathbf{K}_{\mathbf{x}_*}^\top \overline{\mathbf{c}},$$

where $\mathbf{K}_{\mathbf{x}_*} \in \mathbf{R}^{D \times ND}$ has entries $(\mathbf{K}(\mathbf{x}_*, \mathbf{x}_j))_{d,d'}$ for $j = 1, \ldots, N$ and $d, d' = 1, \ldots, D$.

3.3 Gaussian Processes for Vector Valued Functions

Gaussian process methods for modeling vector-valued functions follow the same approach as in the single output case. Recall that a Gaussian process is defined as a collection of random variables, such that any finite number of them follows a joint Gaussian distribution. In the single output case, the random variables are associated to a single process f evaluated at different values of $\mathbf{x}$ while in the multiple output case, the random variables are associated to different processes $\{f_d\}_{d=1}^D$, evaluated at different values of $\mathbf{x}$ [24, 37, 102].

The vector-valued function $\mathbf{f}$ is assumed to follow a Gaussian process

$$\mathbf{f} \sim \mathcal{GP}(\mathbf{m}, \mathbf{K}), \tag{7}$$

where $\mathbf{m} \in \mathbf{R}^D$ is a vector which components are the mean functions $\{m_d(\mathbf{x})\}_{d=1}^D$ of each output and $\mathbf{K}$ is a positive *matrix* valued function as in section 3.2. The entries $(\mathbf{K}(\mathbf{x}, \mathbf{x}'))_{d,d'}$ in the matrix $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ correspond to the covariances between the outputs $f_d(\mathbf{x})$ and $f_{d'}(\mathbf{x}')$ and express the degree of correlation or similarity between them.

For a set of inputs $\mathbf{X}$, the prior distribution over the vector $\mathbf{f}(\mathbf{X})$ is given by

$$\mathbf{f}(\mathbf{X}) \sim \mathcal{N}(\mathbf{m}(\mathbf{X}), \mathbf{K}(\mathbf{X}, \mathbf{X})),$$

where $\mathbf{m}(\mathbf{X})$ is a vector that concatenates the mean vectors associated to the outputs and the covariance matrix $\mathbf{K}(\mathbf{X}, \mathbf{X})$ is the block partitioned matrix in (6). Without loss of generality, we assume the mean vector to be zero.

In a regression context, the likelihood function for the outputs is often taken to be Gaussian distribution, so that

$$p(\mathbf{y}|\mathbf{f}, \mathbf{x}, \Sigma) = \mathcal{N}(\mathbf{f}(\mathbf{x}), \Sigma),$$

where $\Sigma \in \mathbf{R}^{D \times D}$ is a diagonal matrix with elements[3] $\{\sigma_d^2\}_{d=1}^D$.

For a Gaussian likelihood, the predictive distribution and the marginal likelihood can be derived analytically. The predictive distribution for a new vector $\mathbf{x}_*$ is [82]

$$p(\mathbf{f}(\mathbf{x}_*)|\mathbf{S}, \mathbf{f}, \mathbf{x}_*, \boldsymbol{\phi}) = \mathcal{N}\left(\mathbf{f}_*(\mathbf{x}_*), \mathbf{K}_*(\mathbf{x}_*, \mathbf{x}_*)\right), \tag{8}$$

with

$$\begin{aligned}
\mathbf{f}_*(\mathbf{x}_*) &= \mathbf{K}_{\mathbf{x}_*}^\top (\mathbf{K}(\mathbf{X}, \mathbf{X}) + \boldsymbol{\Sigma})^{-1} \overline{\mathbf{y}}, \\
\mathbf{K}_*(\mathbf{x}_*, \mathbf{x}_*) &= \mathbf{K}(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{K}_{\mathbf{x}_*} (\mathbf{K}(\mathbf{X}, \mathbf{X}) + \boldsymbol{\Sigma})^{-1} \mathbf{K}_{\mathbf{x}_*}^\top,
\end{aligned}$$

where $\boldsymbol{\Sigma} = \Sigma \otimes \mathbf{I}_N$, $\mathbf{K}_{\mathbf{x}_*} \in \mathbf{R}^{D \times ND}$ has entries $(\mathbf{K}(\mathbf{x}_*, \mathbf{x}_j))_{d,d'}$ for $j = 1, \ldots, N$ and $d, d' = 1, \ldots, D$, and $\boldsymbol{\phi}$ denotes a possible set of hyperparameters of the covariance function $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ used to compute $\mathbf{K}(\mathbf{X}, \mathbf{X})$ and the variances of the noise for each output $\{\sigma_d^2\}_{d=1}^D$. Again we note that if we are interested into the distribution of the noisy predictions it is easy to see that we simply have to add $\boldsymbol{P}\boldsymbol{\Sigma}$ to the expression of the prediction variance. The above expression for the mean prediction coincides again with the prediction of the estimator derived in the regularization framework.

In the following chapters we describe several possible choices of kernels (covariance function) for multi-output problems. We start in the next chapter with kernel functions that clearly separate the contributions of input and output. We will see later alternative ways to construct kernel functions that interleave both contributions in a non trivial way.

[3] This relation derives from $y_d(\mathbf{x}) = f_d(\mathbf{x}) + \epsilon_d(\mathbf{x})$, for each d, where $\{\epsilon_d(\mathbf{x})\}_{d=1}^D$ are independent white Gaussian noise processes with variance σ_d^2.

4 Separable Kernels and Sum of Separable Kernels

In this chapter we review a special class of multi-output kernel functions that can be formulated as a sum of products between a kernel function for the input space alone, and a kernel function that encodes the interactions among the outputs. We refer to this type of multi-output kernel functions as *separable kernels* and *sum of separable kernels* (SoS kernels).

We consider a class of kernels of the form

$$(\mathbf{K}(\mathbf{x}, \mathbf{x}'))_{d,d'} = k(\mathbf{x}, \mathbf{x}')k_T(d, d'),$$

where k, k_T are scalar kernels on $\mathcal{X} \times \mathcal{X}$ and $\{1, \dots, D\} \times \{1, \dots, D\}$.

Equivalently one can consider the matrix expression

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}')\mathbf{B}, \tag{9}$$

where $\mathbf{B}$ is a $D \times D$ symmetric and positive semi-definite matrix. We call this class of kernels separable since, comparing to (3), we see that the contribution of input and output is decoupled.

In the same spirit a more general class of kernels is given by

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \sum_{q=1}^{Q} k_q(\mathbf{x}, \mathbf{x}')\mathbf{B}_q.$$

For this class of kernels, the kernel matrix associated to a data set $\mathbf{X}$ has a simpler form and can be written as

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \sum_{q=1}^{Q} \mathbf{B}_q \otimes k_q(\mathbf{X}, \mathbf{X}), \tag{10}$$

where $\otimes$ represents the Kronecker product between matrices. We call this class of kernels sum of separable kernels (SoS kernels).

The simplest example of separable kernel is given by setting $k_T(d, d') = \delta_{d,d'}$, where $\delta_{d,d'}$ is the Kronecker delta. In this case $\mathbf{B} = \mathbf{I}_N$, that is all the outputs are treated as being unrelated. In this case the kernel matrix $\mathbf{K}(\mathbf{X}, \mathbf{X})$, associated to some set of data $\mathbf{X}$, becomes block diagonal. Since the off diagonal terms encode output relatedness. We can see that the matrix $\mathbf{B}$ encodes dependencies among the outputs.

The key question is how to choose the scalar kernels $\{k_q\}_{q=1}^{Q}$ and especially how to design, or learn, the matrices $\{\mathbf{B}_q\}_{q=1}^{Q}$. This is the subject we discuss in the next few sections. We will see that one can approach the problem from a regularization point of view, where kernels will be defined by the choice of suitable regularizers, or, from a Bayesian point of view, constructing covariance functions from explicit generative models for the different output components. As it turns out these two points of view are equivalent and allow for two different interpretations of the same class of models.

4.1 Kernels and Regularizers

In this section we largely follow the results in [64, 65, 27] and [7]. A possible way to design multi-output kernels of the form (9) is given by the following result. If $\mathbf{K}$ is given by (9) then is possible to prove that the norm of a function in the corresponding RKHS can be written as

$$\|\mathbf{f}\|_{\mathbf{K}}^2 = \sum_{d,d'=1}^{D} \mathbf{B}_{d,d'}^{\dagger} \langle f_d, f_{d'} \rangle_k , \tag{11}$$

where $\mathbf{B}^{\dagger}$ is the pseudoinverse of $\mathbf{B}$ and $\mathbf{f} = (f_1, \dots, f_D)$. The above expression gives another way to see why the matrix $\mathbf{B}$ encodes the relation among the components. In fact, we can interpret the right hand side in the above expression as a regularizer inducing specific coupling among different tasks $\langle f_t, f_{t'} \rangle_k$ with different weights given by $\mathbf{B}_{d,d'}^{\dagger}$. This result says that any such regularizer induces a kernel of the form (9). We illustrate the above idea with a few examples.

Mixed Effect Regularizer Consider the regularizer given by

$$R(\mathbf{f}) = A_\omega \left(C_\omega \sum_{\ell=1}^{D} \|f_\ell\|_k^2 + \omega D \sum_{\ell=1}^{D} \|f_\ell - \frac{1}{D} \sum_{q=1}^{D} f_q\|_k^2 \right) \tag{12}$$

where $A_\omega = \frac{1}{2(1-\omega)(1-\omega+\omega D)}$ and $C_\omega = (2 - 2\omega + \omega D)$. The above regularizer is composed of two terms: the first is a standard regularization term on the norm of each component of the estimator; the second forces each f_ℓ to be close to the mean estimator across the components, $\overline{f} = \frac{1}{D}\sum_{q=1}^{D} f_q$. The corresponding kernel imposes a common similarity structure between all the output components and the strength of the similarity is controlled by a parameter ω,

$$\mathbf{K}_\omega(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}')(\omega \mathbf{1} + (1-\omega)\mathbf{I}_D) \tag{13}$$

where $\mathbf{1}$ is the $D \times D$ matrix whose entries are all equal to 1, and k is a scalar kernel on the input space $\mathcal{X}$. Setting $\omega = 0$ corresponds to treating all components independently and the possible similarity among them is not exploited. Conversely, $\omega = 1$ is equivalent to assuming that all components are identical and are explained by the same function. By tuning the parameter ω the above kernel interpolates between this two opposites cases. We note that from a Bayesian perspective B is a correlation matrix with all the off-diagonals equal to ω, which means that the output of the Gaussian process are exchangeable.

Cluster Based Regularizer. Another example of regularizer, proposed in [28], is based on the idea of grouping the components into r clusters and enforcing the components in each cluster to be similar. Following [47], let us define the matrix $\mathbf{E}$ as the $D \times r$ matrix, where r is the number of clusters, such that $\mathbf{E}_{\ell,c} = 1$ if the component l belongs to cluster c and 0 otherwise. Then we can compute the $D \times D$ matrix $\mathbf{M} = \mathbf{E}(\mathbf{E}^\top \mathbf{E})^{-1}\mathbf{E}^\top$ such that $\mathbf{M}_{\ell,q} = \frac{1}{m_c}$ if components l and q belong to the same cluster c, and m_c is its cardinality, $\mathbf{M}_{\ell,q} = 0$ otherwise. Furthermore let $I(c)$ be the index set of the components that belong to cluster c. Then we can consider the following regularizer that forces components belonging to the same cluster to be close to each other:

$$R(\mathbf{f}) = \epsilon_1 \sum_{c=1}^{r} \sum_{\ell \in I(c)} \|f_\ell - \overline{f}_c\|_k^2 + \epsilon_2 \sum_{c=1}^{r} m_c \|\overline{f}_c\|_k^2, \tag{14}$$

where $\overline{f}_c$ is the mean of the components in cluster c and ϵ_1, ϵ_2 are parameters balancing the two terms. Straightforward calculations show that the previous regularizer can be rewritten as $R(\mathbf{f}) = \sum_{\ell,q} \mathbf{G}_{\ell,q} \langle f_\ell, f_q \rangle_k$, where

$$\mathbf{G}_{\ell,q} = \epsilon_1 \delta_{lq} + (\epsilon_2 - \epsilon_1)\mathbf{M}_{\ell,q}. \tag{15}$$

Therefore the corresponding matrix valued kernel is $\mathbf{K}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}')\mathbf{G}^\dagger$.

Graph Regularizer. Following [64, 91], we can define a regularizer that, in addition to a standard regularization on the single components, forces stronger or weaker similarity between them through a given $D \times D$ positive weight matrix $\mathbf{M}$,

$$R(\mathbf{f}) = \frac{1}{2} \sum_{\ell,q=1}^{D} \|f_\ell - f_q\|_k^2 \mathbf{M}_{\ell q} + \sum_{\ell=1}^{D} \|f_\ell\|_k^2 \mathbf{M}_{\ell,\ell}. \tag{16}$$

The regularizer $J(f)$ can be rewritten as:

$$\begin{aligned} \sum_{\ell,q=1}^{D} \left(\|f_\ell\|_k^2 \mathbf{M}_{\ell,q} - \langle f_\ell, f_q \rangle_k \mathbf{M}_{\ell,q} \right) + \sum_{\ell=1}^{D} \|f_\ell\|_k^2 \mathbf{M}_{\ell,\ell} &= \\ \sum_{\ell=1}^{D} \|f_\ell\|_k^2 \sum_{q=1}^{D} (1 + \delta_{\ell,q}) \mathbf{M}_{\ell,q} - \sum_{\ell,q=1}^{D} \langle f_\ell, f_q \rangle_k \mathbf{M}_{\ell,q} &= \\ \sum_{\ell,q=1}^{D} \langle f_\ell, f_q \rangle_k \mathbf{L}_{\ell,q} & \end{aligned} \tag{17}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{M}$, with $\mathbf{D}_{\ell,q} = \delta_{\ell,q} \left(\sum_{h=1}^{D} \mathbf{M}_{\ell,h} + \mathbf{M}_{\ell,q} \right)$. Therefore the resulting kernel will be $\mathbf{K}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}')\mathbf{L}^\dagger$, with $k(\mathbf{x}, \mathbf{x}')$ a scalar kernel to be chosen according to the problem at hand.

In the next section we will see how models related to those described above can be derived from suitable generative models.

4.2 Coregionalization Models

The use of probabilistic models and Gaussian processes for multi-output learning was pioneered and largely developed in the context of geostatistics, where prediction over vector-valued output data is known as *cokriging*. Geostatistical approaches to multivariate modelling are mostly formulated around the "linear model of coregionalization" (LMC) [49, 37], that can be considered as a generative approach for developing valid covariance functions. Covariance functions obtained under the LMC assumption follow the form of a sum of separable kernels. We will start considering this model and then discuss how several models recently proposed in the machine learning literature are special cases of the LMC.

4.2.1 The Linear Model of Coregionalization

In the linear model of coregionalization, the outputs are expressed as linear combinations of independent random functions. This is done in a way that ensures that the resulting covariance function (expressed jointly over all the outputs and the inputs) is a valid positive semidefinite function. Consider a set of D outputs $\{f_d(\mathbf{x})\}_{d=1}^D$ with $\mathbf{x} \in \mathbf{R}^p$. In the LMC, each component f_d is expressed as [49]

$$f_d(\mathbf{x}) = \sum_{q=1}^{Q} a_{d,q} u_q(\mathbf{x}),$$

where the latent functions $u_q(\mathbf{x})$, have mean zero and covariance $\text{cov}[u_q(\mathbf{x}), u_{q'}(\mathbf{x}')] = k_q(\mathbf{x}, \mathbf{x}')$ if $q = q'$, and $a_{d,q}$ are scalar coefficients. The processes $\{u_q(\mathbf{x})\}_{q=1}^Q$ are independent for $q \neq q'$. The independence assumption can be relaxed and such relaxation is presented as an extension in section 4.3. Some of the basic processes $u_q(\mathbf{x})$ and $u_{q'}(\mathbf{x}')$ can have the same covariance $k_q(\mathbf{x}, \mathbf{x}')$, while remaining independent.

A similar expression for $\{f_d(\mathbf{x})\}_{d=1}^D$ can be written grouping the functions $u_q(\mathbf{x})$ which share the same covariance [49, 37]

$$f_d(\mathbf{x}) = \sum_{q=1}^{Q} \sum_{i=1}^{R_q} a_{d,q}^i u_q^i(\mathbf{x}), \tag{18}$$

where the functions $u_q^i(\mathbf{x})$, with $q = 1, \ldots, Q$ and $i = 1, \ldots, R_q$, have mean equal to zero and covariance $\text{cov}[u_q^i(\mathbf{x}), u_{q'}^{i'}(\mathbf{x}')] = k_q(\mathbf{x}, \mathbf{x}')$ if $i = i'$ and $q = q'$. Expression (18) means that there are Q groups of functions $u_q^i(\mathbf{x})$ and that the functions $u_q^i(\mathbf{x})$ within each group share the same covariance, but are independent. The cross covariance between any two functions $f_d(\mathbf{x})$ and $f_{d'}(\mathbf{x})$ is given in terms of the covariance functions for $u_q^i(\mathbf{x})$

$$\text{cov}[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] = \sum_{q=1}^{Q} \sum_{q'=1}^{Q} \sum_{i=1}^{R_q} \sum_{i'=1}^{R_q} a_{d,q}^i a_{d',q'}^{i'} \text{cov}[u_q^i(\mathbf{x}), u_{q'}^{i'}(\mathbf{x}')].$$

The covariance $\text{cov}[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')]$ is given by $(\mathbf{K}(\mathbf{x}, \mathbf{x}'))_{d,d'}$. Due to the independence of the functions $u_q^i(\mathbf{x})$, the above expression reduces to

$$(\mathbf{K}(\mathbf{x}, \mathbf{x}'))_{d,d'} = \sum_{q=1}^{Q} \sum_{i=1}^{R_q} a_{d,q}^i a_{d',q}^i k_q(\mathbf{x}, \mathbf{x}') = \sum_{q=1}^{Q} b_{d,d'}^q k_q(\mathbf{x}, \mathbf{x}'), \tag{19}$$

with $b_{d,d'}^q = \sum_{i=1}^{R_q} a_{d,q}^i a_{d',q}^i$. The kernel $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ can now be expressed as

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \sum_{q=1}^{Q} \mathbf{B}_q k_q(\mathbf{x}, \mathbf{x}'), \tag{20}$$

where each $\mathbf{B}_q \in \mathbf{R}^{D \times D}$ is known as a *coregionalization matrix*. The elements of each $\mathbf{B}_q$ are the coefficients $b_{d,d'}^q$ appearing in equation (19). The rank for each matrix $\mathbf{B}_q$ is determined by the number of latent functions that share the same covariance function $k_q(\mathbf{x}, \mathbf{x}')$, that is, by the coefficient R_q.

Equation (18) can be interpreted as a nested structure [104] in which the outputs $f_d(\mathbf{x})$ are first expressed as a linear combination of spatially uncorrelated processes $f_d(\mathbf{x}) = \sum_{q=1}^{Q} f_d^q(\mathbf{x})$, with $\text{E}[f_d^q(\mathbf{x})] = 0$ and $\text{cov}[f_d^q(\mathbf{x}), f_{d'}^{q'}(\mathbf{x}')] = b_{d,d'}^q k_q(\mathbf{x}, \mathbf{x}')$ if $q = q'$, otherwise it is equal to zero. At the same time, each process $f_d^q(\mathbf{x})$ can be represented as a set

of uncorrelated functions weighted by the coefficients $a_{d,q}^i$, $f_d^q(\mathbf{x}) = \sum_{i=1}^{R_q} a_{d,q}^i u_q^i(\mathbf{x})$ where again, the covariance function for $u_q^i(\mathbf{x})$ is $k_q(\mathbf{x}, \mathbf{x}')$.

Therefore, starting from a generative model for the outputs, the linear model of coregionalization leads to a sum of separable kernels that represents the covariance function as the sum of the products of two covariance functions, one that models the dependence between the outputs, independently of the input vector $\mathbf{x}$ (the coregionalization matrix $\mathbf{B}_q$), and one that models the input dependence, independently of the particular set of functions $\{f_d(\mathbf{x})\}$ (the covariance function $k_q(\mathbf{x}, \mathbf{x}')$). The covariance matrix for $\mathbf{f}(\mathbf{X})$ is given by (10).

4.2.2 Intrinsic Coregionalization Model

A simplified version of the LMC, known as the intrinsic coregionalization model (ICM) (see [37]), assumes that the elements $b_{d,d'}^q$ of the coregionalization matrix $\mathbf{B}_q$ can be written as $b_{d,d'}^q = \upsilon_{d,d'} b_q$, for some suitable coefficients $\upsilon_{d,d'}$. With this form for $b_{d,d'}^q$, we have

$$\begin{aligned}\operatorname{cov}[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] &= \sum_{q=1}^{Q} \upsilon_{d,d'} b_q k_q(\mathbf{x}, \mathbf{x}'), = \upsilon_{d,d'} \sum_{q=1}^{Q} b_q k_q(\mathbf{x}, \mathbf{x}') \\ &= \upsilon_{d,d'} k(\mathbf{x}, \mathbf{x}'),\end{aligned}$$

where $k(\mathbf{x}, \mathbf{x}') = \sum_{q=1}^{Q} b_q k_q(\mathbf{x}, \mathbf{x}')$. The above expression can be seen as a particular case of the kernel function obtained from the linear model of coregionalization, with $Q = 1$. In this case, the coefficients $\upsilon_{d,d'} = \sum_{i=1}^{R_1} a_{d,1}^i a_{d',1}^i = b_{d,d'}^1$, and the kernel matrix for multiple outputs becomes $\mathbf{K}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}')\mathbf{B}$ as in (9).

The kernel matrix corresponding to a dataset $\mathbf{X}$ takes the form

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}). \tag{21}$$

One can see that the intrinsic coregionalization model corresponds to the special separable kernel often used in the context of regularization. Notice that the value of R_1 for the coefficients $\upsilon_{d,d'} = \sum_{i=1}^{R_1} a_{d,1}^i a_{d',1}^i = b_{d,d'}^1$, determines the rank of the matrix $\mathbf{B}$.

As pointed out by [37], the ICM is much more restrictive than the LMC since it assumes that each basic covariance $k_q(\mathbf{x}, \mathbf{x}')$ contributes equally to the construction of the autocovariances and cross covariances for the outputs. However, the computations required for the corresponding inference are greatly simplified, essentially because of the properties of the Kronecker product. This latter point is discussed in detail in Section 6.

It can be shown that if the outputs are considered to be noise-free, prediction using the intrinsic coregionalization model under an isotopic data case is equivalent to independent prediction over each output [41]. This circumstance is also known as autokrigeability [104].

4.2.3 Comparison Between ICM and LMC

We have seen before that the intrinsic coregionalization model is a particular case of the linear model of coregionalization for $Q = 1$ (with $R_q \neq 1$) in equation 19. Here we contrast these two models. Note that a different particular case of the linear model of coregionalization is assuming $R_q = 1$ (with $Q \neq 1$). This model, known in the machine learning literature as the semiparametric latent factor model (SLFM) [96], will be introduced in the next subsection.

To compare the two models we have sampled from a multi-output Gaussian process with two outputs ($D = 2$), a one-dimensional input space ($x \in \mathbf{R}$) and a LMC with different values for R_q and Q. As basic kernels $k_q(\mathbf{x}, \mathbf{x}')$ we have used the exponentiated quadratic (EQ) kernel given as [82],

$$k_q(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\ell_q^2}\right),$$

where $\|\cdot\|$ represents the Euclidian norm and ℓ_q is known as the characteristic length-scale. The exponentiated quadratic is variously referred to as the Gaussian, the radial basis function or the squared exponential kernel.

Figure 2 shows samples from the intrinsic coregionalization model for $R_q = 1$, meaning a coregionalization matrix $\mathbf{B}_1$ of rank one. Samples share the same length-scale and have similar form. They have different variances, though. Each sample may be considered as a scaled version of the latent function, as it can be seen from equation 18 with $Q = 1$ and $R_q = 1$,

$$f_1(x) = a_{1,1}^1 u_1^1(x), \qquad f_2(x) = a_{2,1}^1 u_1^1(x),$$

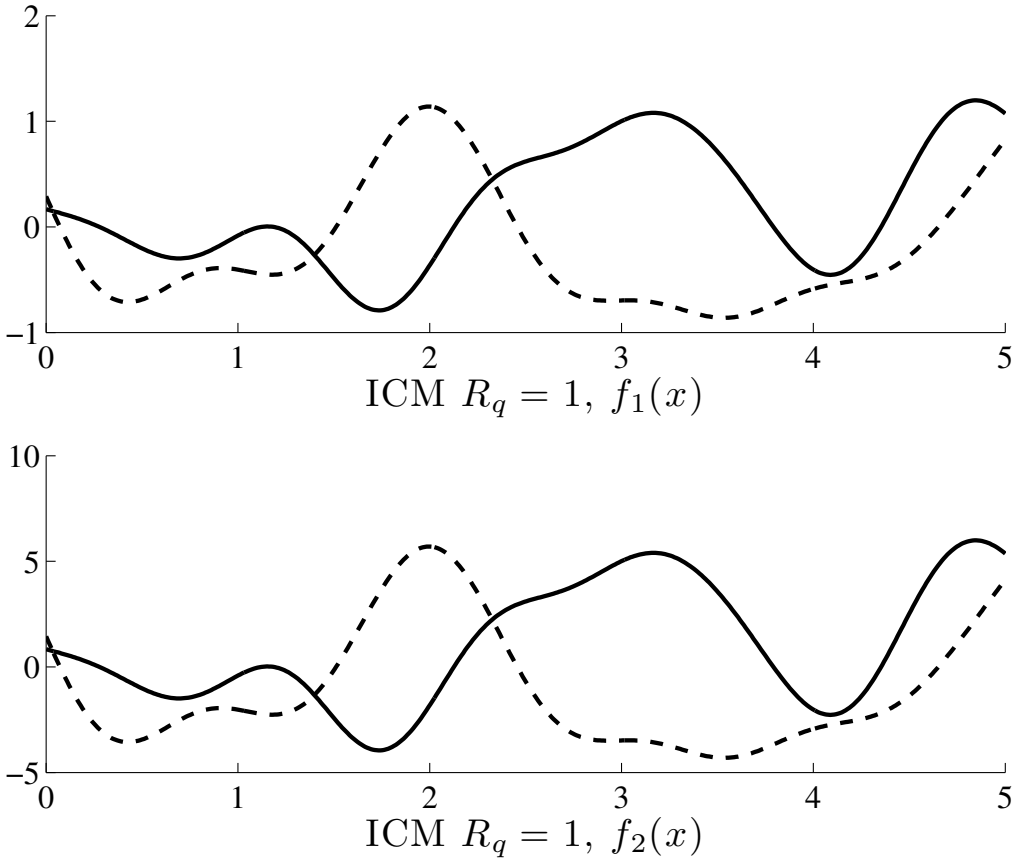

Figure 2: Two samples from the intrinsic coregionalization model with rank one, this is $R_q = 1$. Solid lines represent one of the samples, and dashed lines represent the other sample. Samples are identical except for scale.

where we have used x instead of $\mathbf{x}$ for the one-dimensional input space.

Figure 3 shows samples from an ICM of rank two. From equation 18, we have for $Q = 1$ and $R_q = 2$,

$$f_1(x) = a^1_{1,1}u^1_1(x) + a^2_{1,1}u^2_1(x), \qquad f_2(x) = a^1_{2,1}u^1_1(x) + a^2_{2,1}u^2_1(x),$$

where $u^1_1(x)$ and $u^2_1(x)$ are sampled from the same Gaussian process. Outputs are weighted sums of two different latent functions that share the same covariance. In contrast to the ICM of rank one, we see from figure 3 that both outputs have different forms, although they share the same length-scale.

Figure 4 displays outputs sampled from a LMC with $R_q = 1$ and two latent functions ($Q = 2$) with different length-scales. Notice that both samples are combinations of two terms, a long length-scale term and a short length-scale term. According to equation 18, outputs are given as

$$f_1(x) = a^1_{1,1}u^1_1(x) + a^1_{1,2}u^1_2(x), \qquad f_2(x) = a^1_{2,1}u^1_1(x) + a^1_{2,2}u^1_2(x),$$

where $u^1_1(x)$ and $u^1_2(x)$ are samples from two Gaussian processes with different covariance functions. In a similar way to the ICM of rank one (see figure 2), samples from both outputs have the same form, this is, they are aligned.

We have the additional case for a LMC with $R_q = 2$ and $Q = 2$ in figure 5. According to equation 18, the outputs are give as

$$f_1(x) = a^1_{1,1}u^1_1(x) + a^2_{1,1}u^2_1(x) + a^1_{1,2}u^1_2(x) + a^2_{1,2}u^2_2(x),$$
$$f_2(x) = a^1_{2,1}u^1_1(x) + a^2_{2,1}u^2_1(x) + a^1_{2,2}u^1_2(x) + a^2_{2,2}u^2_2(x),$$

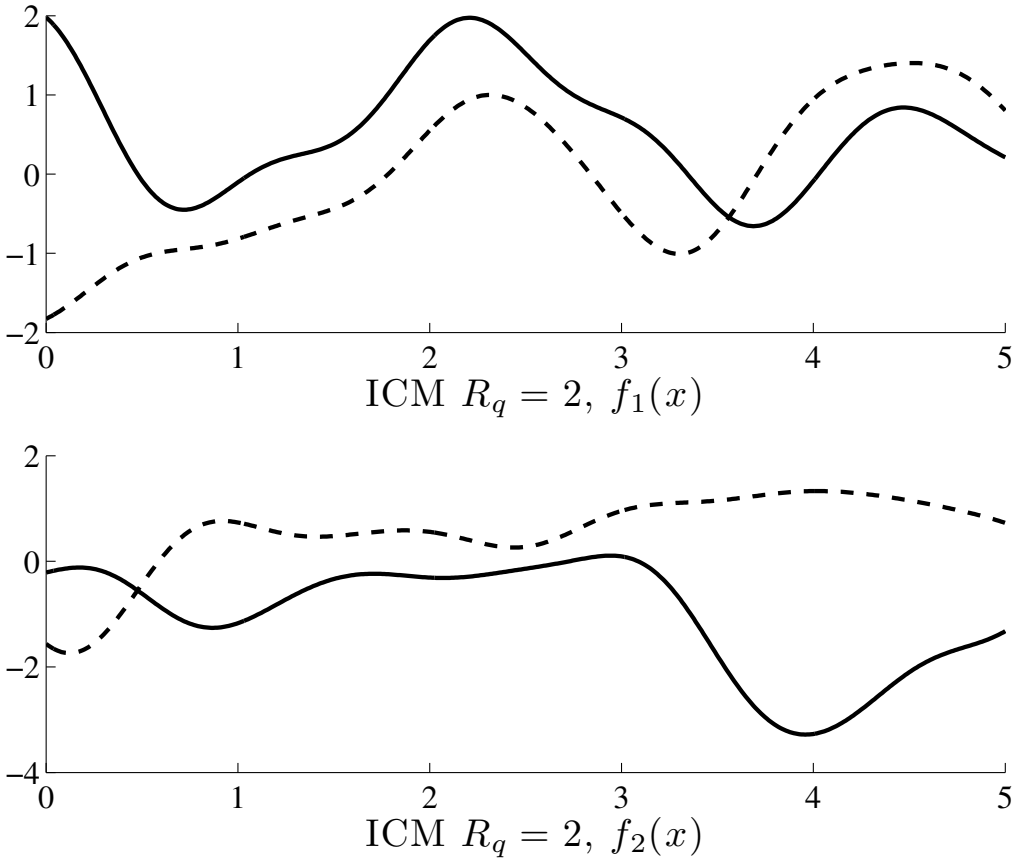

Figure 3: Two samples from the intrinsic coregionalization model with rank two, $R_q = 2$. Solid lines and dashed lines represent different samples. Although samples from different outputs have the same length-scale, they look different and are not simply scaled versions of one another.

where the pair of latent functions $u_1^1(x)$ and $u_1^2(x)$ share their covariance function and the pair of latent functions $u_2^1(x)$ and $u_2^2(x)$ also share their covariance function. As in the case of the LMC with $R_q = 1$ and $Q = 2$ in figure 4, the outputs are combinations of a term with a long length-scale and a term with a short length-scale. A key difference however, is that, for $R_q = 2$ and $Q = 2$, samples from different outputs have different shapes.[4]

4.2.4 Linear Model of Coregionalization in Machine Learning and Statistics

The linear model of coregionalization has already been used in machine learning in the context of Gaussian processes for multivariate regression and in statistics for computer emulation of expensive multivariate computer codes.

As we have seen before, the linear model of coregionalization imposes the correlation of the outputs explicitly through the set of coregionalization matrices. A simple idea used in the early papers of multi-output GPs for machine learning was based on the intrinsic coregionalization model and assumed $\mathbf{B} = \mathbf{I}_D$. In other words, the outputs were considered to be conditionally independent given the parameters $\boldsymbol{\phi}$. Correlation between the outputs was assumed to exist implicitly by imposing the same set of hyperparameters $\boldsymbol{\phi}$ for all outputs and estimating those parameters, or the kernel matrix $k(\mathbf{X}, \mathbf{X})$ directly, using data from all the outputs [66, 55, 113].

[4]Notice that samples from each output are not synchronized, meaning that the maximums and minimus do not always occur at the same input points.

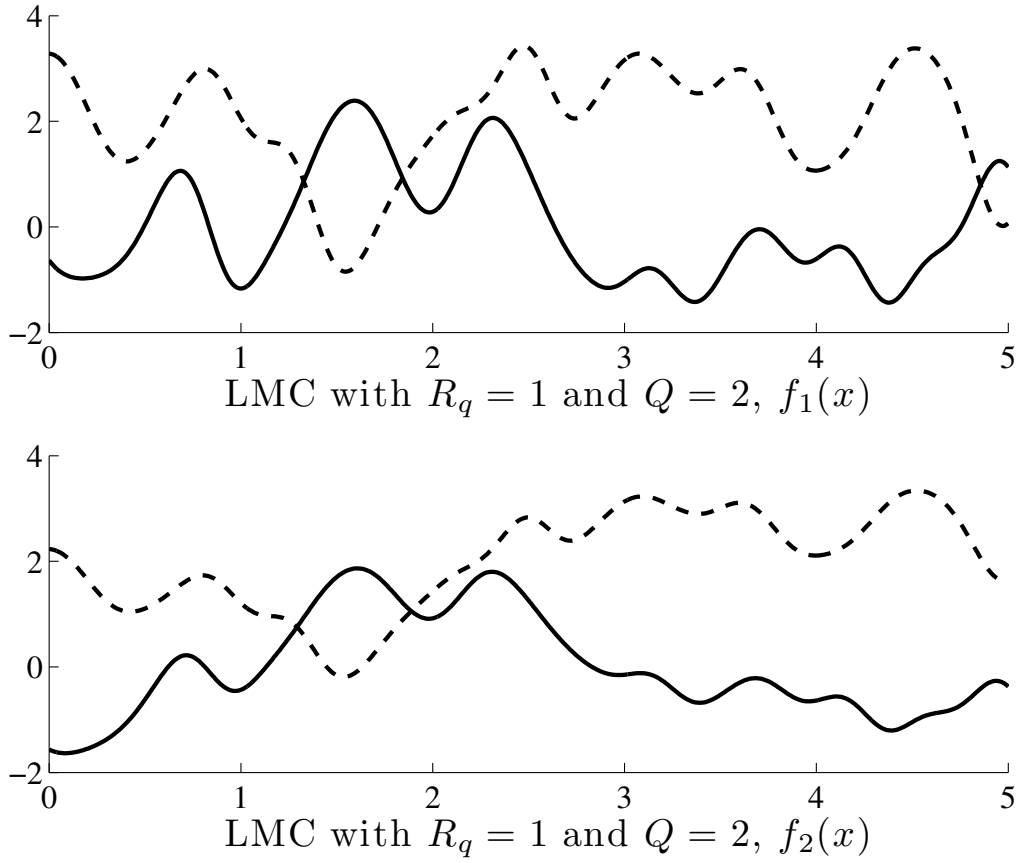

Figure 4: Two samples from a linear model of coregionalization with $R_q = 1$ and $Q = 2$. The solid lines represent one of the samples. The dashed lines represent the other sample. Samples are the weigthed sums of latent functions with different length-scales.

In this section, we review more recent approaches for multiple output modeling that are different versions of the linear model of coregionalization.

Semiparametric latent factor model. The semiparametric latent factor model (SLFM) proposed by [96] turns out to be a simplified version of the LMC. In fact it corresponds to setting $R_q = 1$ in (18) so that we can rewrite equation (10) as

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \sum_{q=1}^{Q} \mathbf{a}_q \mathbf{a}_q^\top \otimes k_q(\mathbf{X}, \mathbf{X}),$$

where $\mathbf{a}_q \in \mathbf{R}^{D\times 1}$ with elements $\{a_{d,q}\}_{d=1}^{D}$ and q fixed. With some algebraic manipulations, that exploit the properties of the Kronecker product, we can write

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \sum_{q=1}^{Q} (\mathbf{a}_q \otimes \mathbf{I}_N) k_q(\mathbf{X}, \mathbf{X}) (\mathbf{a}_q^\top \otimes \mathbf{I}_N) = (\widetilde{\mathbf{A}} \otimes \mathbf{I}_N) \widetilde{\mathbf{K}} (\widetilde{\mathbf{A}}^\top \otimes \mathbf{I}_N),$$

where $\widetilde{\mathbf{A}} \in \mathbf{R}^{D\times Q}$ is a matrix with columns $\mathbf{a}_q$ and $\widetilde{\mathbf{K}} \in \mathbf{R}^{QN\times QN}$ is a block diagonal matrix with blocks given by $k_q(\mathbf{X}, \mathbf{X})$.

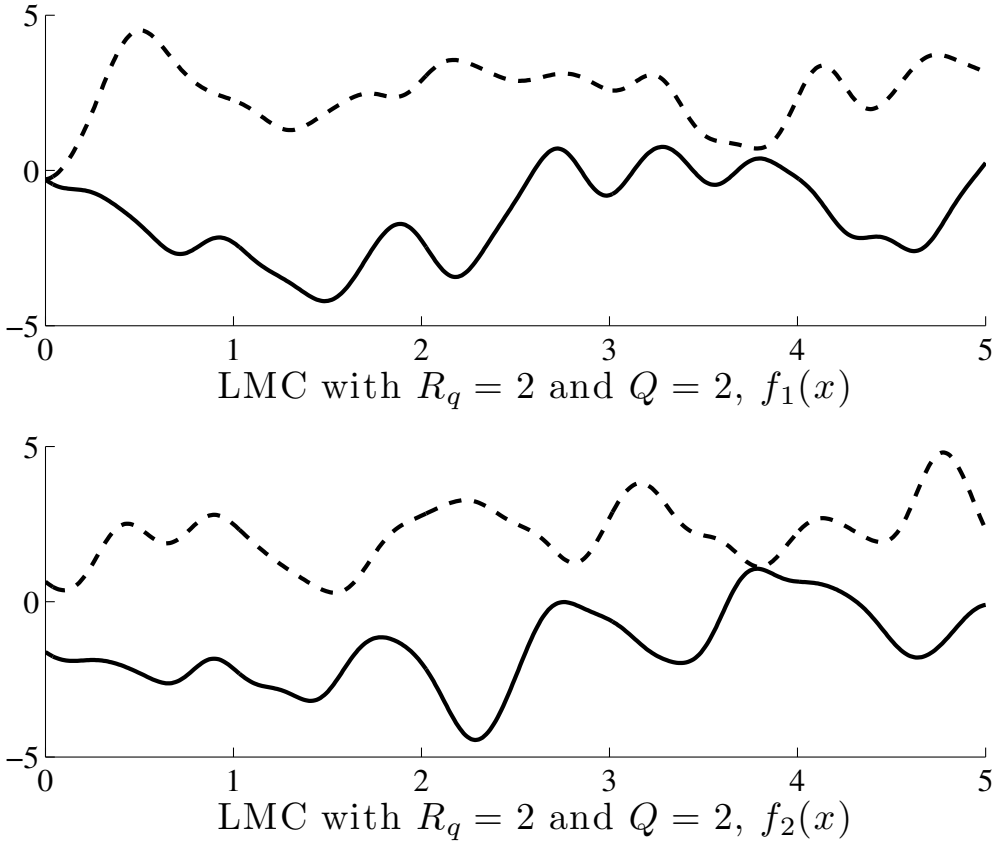

Figure 5: Two samples from a linear model of coregionalization with $R_q = 2$ and $Q = 2$. The solid lines represent one of the samples. The dashed lines represent the other sample. Samples are the weigthed sums of four latent functions, two of them share a covariance with a long length-scale and the other two share a covariance with a shorter length-scale.

The functions $u_q(\mathbf{x})$ are considered to be latent factors and the semiparametric name comes from the fact that it is combining a nonparametric model, that is a Gaussian process, with a parametric linear mixing of the functions $u_q(\mathbf{x})$. The kernels k_q, for each basic process is assumed to be exponentiated quadratic with a different characteristic length-scale for each input dimension. The informative vector machine (IVM) [57] is employed to speed up computations.

Gaussian processes for Multi-task, Multi-output and Multi-class The intrinsic coregionalization model is considered by [12] in the context of multitask learning. The authors use a probabilistic principal component analysis (PPCA) model to represent the matrix $\mathbf{B}$. The spectral factorization in the PPCA model is replaced by an incomplete Cholesky decomposition to keep numerical stability. The authors also refer to the autokrigeability effect as the cancellation of inter-task transfer [12], and discuss the similarities between the multi-task GP and the ICM, and its relationship to the SLFM and the LMC.

The intrinsic coregionalization model has also been used by [72]. Here the matrix $\mathbf{B}$ is assumed to have a spherical parametrization, $\mathbf{B} = \mathrm{diag}(\mathbf{e})\mathbf{S}^\top\mathbf{S}\,\mathrm{diag}(\mathbf{e})$, where $\mathbf{e}$ gives a description for the scale length of each output variable and $\mathbf{S}$ is an upper triangular matrix whose i-th column is associated with particular spherical coordinates of points in $\mathbf{R}^i$ (for details see sec. 3.4 [71]). The scalar kernel k is represented through a Matérn kernel, where different parameterizations allow the expression of periodic and non-periodic terms. Sparsification for this model is obtained using an IVM style approach.

In a classification context, Gaussian processes methodology has been mostly restricted to the case where the

outputs are conditionally independent given the hyperparameters ϕ [66, 110, 55, 89, 113, 82]. Therefore, the kernel matrix $\mathbf{K}(\mathbf{X}, \mathbf{X})$ takes a block-diagonal form, with blocks given by $(\mathbf{K}(\mathbf{X}_d, \mathbf{X}_d))_{d,d}$. Correlation between the outputs is assumed to exist implicitly by imposing the same set of hyperparameters ϕ for all outputs and estimating those parameters, or directly the kernel matrices $(\mathbf{K}(\mathbf{X}_d, \mathbf{X}_d))_{d,d}$, using data from all the outputs [66, 55, 113, 82]. Alternatively, it is also possible to have parameters ϕ_d associated to each output [110, 89].

Only recently, the intrinsic coregionalization model has been used in the multiclass scenario. In [93], the authors use the intrinsic coregionalization model for classification, by introducing a probit noise model as the likelihood. Since the posterior distribution is no longer analytically tractable, the authors use Gibbs sampling, Expectation-Propagation (EP) and variational Bayes [5] to approximate the distribution.

Computer emulation. A computer emulator is a statistical model used as a surrogate for a computationally expensive deterministic model or computer code, also known as a simulator. Gaussian processes have become the preferred statistical model among computer emulation practitioners (for a review see [70]). Different Gaussian process emulators have been recently proposed to deal with several outputs [42, 23, 83, 63, 9, 79].

In [42], the linear model of coregionalization is used to model images representing the evolution of the implosion of steel cylinders after using TNT and obtained employing the so called Neddemeyer simulation model (see [42] for further details). The input variable $\mathbf{x}$ represents parameters of the simulation model, while the output is an image of the radius of the inner shell of the cylinder over a fixed grid of times and angles. In the version of the LMC that the authors employed, $R_q = 1$ and the Q vectors $\mathbf{a}_q$ were obtained as the eigenvectors of a PCA decomposition of the set of training images.

In [23], the intrinsic coregionalization model is employed for emulating the response of a vegetation model called the Sheffield Dynamic Global Vegetation Model (SDGVM) [111]. Authors refer to the ICM as the Multiple-Output (MO) emulator. The inputs to the model are ten ($p = 10$) variables related to broad soil, vegetation and climate data, while the outputs are time series of the net biome productivity (NBP) index measured at a particular site in a forest area of Harwood, UK. The NBP index accounts for the residual amount of carbon at a vegetation site after some natural processes have taken place. In the paper, the authors assume that the outputs correspond to the different sampling time points, so that $D = T$, being T the number of time points, while each observation corresponds to specific values of the ten input variables. Values of the input variables are chosen according to a maxi-min Latin hypercube design.

Rougier [83] introduces an emulator for multiple-outputs that assumes that the set of output variables can be seen as a single variable while augmenting the input space with an additional index over the outputs. In other words, it considers the output variable as an input variable. [23], refers to the model in [83] as the Time Input (TI) emulator and discussed how the TI model turns out to be a particular case of the MO model that assumes a particular exponentiated quadratic kernel (see chapter 4 [82]) for the entries in the coregionalization matrix $\mathbf{B}$.

McFarland *et al.* [63] consider a multiple-output problem as a single output one. The setup is similar to the one used in [23], where the number of outputs are associated to different time points, this is, $D = T$. The outputs correspond to the time evolutions of the temperature of certain location of a container with decomposing foam, as function of five different *calibration* variables (input variables in this context, $p = 5$). The authors use the time index as an input (akin to [83]) and apply a greedy-like algorithm to select the training points for the Gaussian process. Greedy approximations like this one have also been used in the machine learning literature (for details, see [82], page 174).

Similar to [83] and [63], Bayarri *et al.* [9] use the time index as an input for a computer emulator that evaluates the accuracy of CRASH, a computer model that simulates the effect of a collision of a vehicle with different types of barriers.

Quian *et al.* [79] propose a computer emulator based on Gaussian processes that supports quantitative and qualitative inputs. The covariance function in this computer emulator is related to the ICM in the case of one qualitative factor: the qualitative factor is considered to be the index of the output, and the covariance function takes again the form $k(\mathbf{x}, \mathbf{x}')k_T(d, d')$. In the case of more than one qualitative input, the computer emulator could be considered a multiple output GP in which each output index would correspond to a particular combination of the possible values taken by the qualitative factors. In this case, the matrix $\mathbf{B}$ in ICM would have a block diagonal form, each block determining the covariance between the values taken by a particular qualitative input.

[5] Mathematical treatment for each of these inference methods can be found in [10], chapters 10 and 11.

4.3 Extensions

In this section we describe further developments related to the setting of separable kernels or SoS kernels, both from a regularization and a Bayesian perspective.

4.3.1 Extensions Within the Regularization Framework

When we consider kernels of the form $\mathbf{K}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}')\mathbf{B}$, a natural question is whether the matrix $\mathbf{B}$ can be learned from data. In a regression setting, one idea is to estimate $\mathbf{B}$ in a separate inference step as the covariance matrix of the output vectors in the training set and this is standard in the geostatistics literature [104]. A further question is whether we can learn both $\mathbf{B}$ and an estimator within a unique inference step. This is the question tackled in [48]. The authors consider a variation of the regularizer in (14) and try to learn the cluster matrix as a part of the optimization process. More precisely the authors considered a regularization term of the form

$$R(\mathbf{f}) = \epsilon_1 \|\overline{f}\|_k + \epsilon_2 \sum_{c=1}^{r} m_c \|\overline{f}_c - \overline{f}\|_k^2 + \epsilon_3 \sum_{c=1}^{r} \sum_{l \in I(c)} \|f^l - \overline{f}_c\|_k^2, \tag{22}$$

where we recall that r is the number of clusters. The three terms in the functional can be seen as: a global penalty, a term penalizing *between cluster* variance and a term penalizing *within cluster* variance. As in the case of the regularizer in (14), the above regularizer is completely characterized by a cluster matrix $\mathbf{M}$, i.e. $R(\mathbf{f}) = R_{\mathbf{M}}(\mathbf{f})$ (note that the corresponding matrix $\mathbf{B}$ will be slightly different from (15)).

The idea is then to consider a regularized functional

$$\sum_{i=1}^{D} \frac{1}{N} \sum_{i=1}^{N} (f_j(\mathbf{x}_i) - y_{j,i})^2 + \lambda R_{\mathbf{M}}(\mathbf{f}) \tag{23}$$

to be minimized jointly over $\mathbf{f}$ and $\mathbf{M}$ (see [48] for details). This problem is typically non tractable from a computational point of view, so the authors in [48] propose a relaxation of the problem which can be shown to be convex.

A different approach is taken in [4] and [5]. In this case the idea is that only a a small subset of features is useful to learn all the components/tasks. In the simplest case the authors propose to minimize a functional of the form

$$\sum_{d=1}^{D} \left\{ \frac{1}{N} \sum_{i=1}^{N} (\mathbf{w}_d^\top U^\top \mathbf{x}_i - y_{d,i})^2 + \lambda \mathbf{w}_d^\top \mathbf{w}_d \right\}.$$

over $\mathbf{w}_1, \ldots, \mathbf{w}_D \in \mathbf{R}^p, U \in \mathbf{R}^{D \times D}$ under the constraint $\mathrm{Tr}(U_t^\top U_t) \leq \gamma$. Note that the minimization over the matrix U couples the otherwise disjoint component-wise problems. The authors of [4] discuss how the above model is equivalent to considering a kernel of the form

$$K(\mathbf{x}, \mathbf{x}') = k_{\mathbf{D}}(\mathbf{x}, \mathbf{x}')\mathbf{I}_D, \qquad k_{\mathbf{D}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{D} \mathbf{x}'$$

where $\mathbf{D}$ is a positive definite matrix and a model which can be described components wise as

$$f_d(\mathbf{x}) = \sum_{i=1}^{p} a_{d,i} x_j = \mathbf{a}_d^\top \mathbf{x},$$

making apparent the connection with the LMC model. In fact, it is possible to show that the above minimization problem is equivalent to minimizing

$$\sum_{d=1}^{D} \frac{1}{N} \sum_{i=1}^{N} (\mathbf{a}_d^\top \mathbf{x}_i - y_{d,i})^2 + \lambda \sum_{d=1}^{D} \mathbf{a}_d^\top \mathbf{D} \mathbf{a}_d, \tag{24}$$

over $\mathbf{a}'_1, \ldots, \mathbf{a}'_D \in \mathbf{R}^p$ and $\mathrm{Tr}(\mathbf{D}) \leq 1$, where the last restriction is a convex approximation of the low rank requirement. Note that from a Bayesian perspective the above scheme can be interpreted as learning a covariance matrix for the response variables which is optimal for all the tasks. In [4], the authors consider a more general setting where $\mathbf{D}$ is replaced by $F(\mathbf{D})$ and show that if the matrix valued function F is matrix concave, then the induced minimization problem is jointly convex in $(\mathbf{a}_i)$ and $\mathbf{D}$. Moreover, the authors discuss how to extend the

above framework to the case of more general kernel functions. Note that an approach similar to the one we just described is at the basis of recent work exploiting the concept of sparsity while solving multiple tasks. These latter methods cannot in general be cast in the framework of kernel methods and we refer the interested reader to [69] and references therein.

For the reasoning above the key assumption is that a response variable is either important for all the tasks or not. In practice it is probably often the case that only certain subgroups of tasks share the same variables. This idea is at the basis of the study in [5], where the authors design an algorithm to learn at once the group structure and the best set of variables for each groups of tasks. Let $\mathcal{G} = (G_t)_{t=1}^{\top}$ be a partition of the set of components/tasks, where G_t denotes a group of tasks and $|G_t| \leq D$. Then the author propose to consider a functional of the form

$$\min_{\mathcal{G}} \sum_{G_t \in \mathcal{G}} \min_{\mathbf{a}_d, d \in G_t, U_t} \sum_{d \in G_t} \left\{ \frac{1}{N} \sum_{i=1}^{N} (\mathbf{a}_d^{\top} U_t^{\top} \mathbf{x}_i - y_{d,i})^2 + \lambda \mathbf{w}_d^{\top} \mathbf{w}_d^{\top} + \gamma \mathrm{Tr}(U_t^{\top} U_t) \right\},$$

where $U_1, \ldots U_T$ is a sequence of p by p matrices. The authors show that while the above minimization problem is not convex, stochastic gradient descent can be used to find local minimizers which seems to perform well in practice.

4.3.2 Extensions from the Gaussian Processes Perspective

A recent extension of the linear model of coregionalization expresses the output covariance function through a linear combination of nonorthogonal latent functions [39]. In particular, the basic processes $u_q^i(\mathbf{x})$ are assumed to be nonorthogonal, leading to the following covariance function

$$\mathrm{cov}[\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')] = \sum_{q=1}^{Q} \sum_{q'=1}^{Q} \mathbf{B}_{q,q'} k_{q,q'}(\mathbf{x}, \mathbf{x}'),$$

where $\mathbf{B}_{q,q'}$ are *cross-coregionalization* matrices. Cross-covariances $k_{q,q'}(\mathbf{x}, \mathbf{x}')$ can be negative (while keeping positive semidefiniteness for $\mathrm{cov}[\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')]$), allowing negative cross-covariances in the linear model of coregionalization. The authors argue that, in some real scenarios, a linear combination of several correlated attributes are combined to represent a single model and give examples in mining, hydrology and oil industry [39].

5 Beyond Separable Kernels

Working with separable kernels or SoS kernels is appealing for their simplicity, but can be limiting in several applications. Next we review different types of kernels that go beyond the separable case or SoS case.

5.1 Invariant Kernels

Divergence free and curl free fields. The following two kernels are matrix valued exponentiated quadratic (EQ) kernels [68] and can be used to estimate divergence-free or curl-free vector fields [60] when the input and output space have the same dimension. These kernels induce a similarity between the vector field components that depends on the input points, and therefore cannot be reduced to the form $\mathbf{K}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}')\mathbf{B}$.

We consider the case of vector fields with $D = p$, where $\mathcal{X} = R^p$. The divergence-free matrix-valued kernel can be defined via a translation invariant matrix-valued EQ kernel

$$\Phi(u) = (\nabla\nabla^{\top} - \nabla^{\top}\nabla I)\phi(u) = H\phi(u) - \mathrm{tr}(H\phi(u))\mathbf{I}_D \ ,$$

where H is the Hessian operator and ϕ a scalar EQ kernel, so that $\mathbf{K}(\mathbf{x}, \mathbf{x}') := \Phi(\mathbf{x} - \mathbf{x}')$.

The columns of the matrix valued EQ kernel, Φ, are divergence-free. In fact, computing the divergence of a linear combination of its columns, $\nabla^{\top}(\Phi(u)c)$, with $c \in \mathbf{R}^p$, it is possible to show that [7]

$$\nabla^{\top}(\Phi(u)c) = (\nabla^{\top}\nabla\nabla^{\top}\phi(u))c - (\nabla^{\top}\nabla^{\top}\nabla\phi(u))c = 0 \ ,$$

where the last equality follows applying the product rule of the gradient, the fact that the coefficient vector c does not depend upon u and the equality $a^\top a a^\top = a^\top a^\top a, \forall a \in \mathbf{R}^p$.

Choosing a exponentiated quadratic, we obtain the divergence-free kernel

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \frac{1}{\sigma^2} e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}} A_{\mathbf{x},\mathbf{x}'}, \tag{25}$$

where

$$A_{\mathbf{x},\mathbf{x}'} = \left(\left(\frac{\mathbf{x}-\mathbf{x}'}{\sigma} \right) \left(\frac{\mathbf{x}-\mathbf{x}'}{\sigma} \right)^\top + \left((p-1) - \frac{\|\mathbf{x}-\mathbf{x}'\|^2}{\sigma^2} \right) \mathbf{I}_p \right) .$$

The curl-free matrix valued kernels are obtained as

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') := \Psi(\mathbf{x}-\mathbf{x}') = -\nabla\nabla^\top \phi(\mathbf{x}-\mathbf{x}') = -H\phi(\mathbf{x}-\mathbf{x}') ,$$

where ϕ is a scalar RBF. It is easy to show that the columns of Ψ are curl-free. The j-th column of Ψ is given by $\Psi \mathbf{e}_j$, where $\mathbf{e}_j$ is the standard basis vector with a one in the j-th position. This gives us

$$\Phi_{cf} e_j = -\nabla\nabla^\top \Phi_{cf} e_j = \nabla(-\nabla^\top \Phi_{cf} e_j) = \nabla g ,$$

where $g = -\partial\phi/\partial x_j$. The function g is a scalar function and the curl of the gradient of a scalar function is always zero. Choosing a exponentiated quadratic, we obtain the following curl-free kernel

$$\Gamma_{cf}(\mathbf{x}, \mathbf{x}') = \frac{1}{\sigma^2} e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}} \left(\mathbf{I}_D - \left(\frac{\mathbf{x}-\mathbf{x}'}{\sigma} \right) \left(\frac{\mathbf{x}-\mathbf{x}'}{\sigma} \right)^\top \right) . \tag{26}$$

It is possible to consider a convex linear combination of these two kernels to obtain a kernel for learning any kind of vector field, while at the same time allowing reconstruction of the divergence-free and curl-free parts separately (see [60]). The interested reader can refer to [68, 59, 32] for further details on matrix-valued RBF and the properties of divergence-free and curl-free kernels.

Transformable kernels. Another example of invariant kernels is discussed in [18] and is given by kernels defined by transformations. For the purpose of our discussion, let $\mathcal{Y} = \mathbf{R}^D$, $\mathcal{X}_0$ be a Hausdorff space and T_d a family of maps (not necessarily linear) from $\mathcal{X}$ to $\mathcal{X}_0$ for $d = \{1, \ldots, D\}$.

Then, given a continuous scalar kernel $k : \mathcal{X}_0 \times \mathcal{X}_0 \to \mathbf{R}$, it is possible to define the following matrix valued kernel for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$

$$\Big(\mathbf{K}(\mathbf{x}, \mathbf{x}')\Big)_{d,d'} = k(T_d\mathbf{x}, T_{d'}\mathbf{x}').$$

A specific instance of the above example is described by [101] in the context of system identification, see also [18] for further details.

5.2 Further Extensions of the LMC

In [34], the authors introduced a nonstationary version of the LMC, in which the coregionalization matrices are allowed to vary as functions of the input variables. In particular, $\mathbf{B}_q$ now depends on the input variable $\mathbf{x}$, this is, $\mathbf{B}_q(\mathbf{x}, \mathbf{x}')$. The authors refer to this model as the spatially varying LMC (SVLMC). As a model for the varying coregionalization matrix $\mathbf{B}_q(\mathbf{x}, \mathbf{x}')$, the authors employ two alternatives. In one of them, they assume that $\mathbf{B}_q(\mathbf{x}, \mathbf{x}') = (\boldsymbol{\alpha}(\mathbf{x}, \mathbf{x}'))^\psi \mathbf{B}_q$, where $\boldsymbol{\alpha}(\mathbf{x})$ is a covariate of the input variable, and ψ is a variable that follows a uniform prior. In the other alternative, $\mathbf{B}_q(\mathbf{x}, \mathbf{x}')$ follows a Wishart spatial process, which is constructed using the definition of a Wishart distribution, as follows. Suppose $\mathbf{Z} \in \mathbf{R}^{D\times P}$ is a random matrix with entries $z_{d,p} \sim \mathcal{N}(0, 1)$, independently and identically distributed, for $d = 1, \ldots, D$ and $p = 1, \ldots, P$. Define the matrix $\boldsymbol{\Upsilon} = \boldsymbol{\Gamma}\mathbf{Z}$, with $\boldsymbol{\Gamma} \in \mathbf{R}^{D\times D}$. The matrix $\boldsymbol{\Omega} = \boldsymbol{\Upsilon}\boldsymbol{\Upsilon}^\top = \boldsymbol{\Gamma}\mathbf{Z}\mathbf{Z}^\top\boldsymbol{\Gamma}^\top \in \mathbf{R}^{D\times D}$ follows a Wishart distribution $\mathcal{W}(P, \boldsymbol{\Gamma}\boldsymbol{\Gamma}^\top)$, where P is known as the *number of degrees of freedom* of the distribution. The spatial Wishart process is constructed assuming that $z_{d,p}$ depends on the input $\mathbf{x}$, this is, $z_{d,p}(\mathbf{x}, \mathbf{x}')$, with $z_{d,p}(\mathbf{x}, \mathbf{x}') \sim \mathcal{N}(0, \rho_d(\mathbf{x}, \mathbf{x}'))$, where $\{\rho_d(\mathbf{x}, \mathbf{x}')\}_{d=1}^D$ are correlation functions. Matrix $\boldsymbol{\Upsilon}(\mathbf{x}, \mathbf{x}') = \boldsymbol{\Gamma}\mathbf{Z}(\mathbf{x}, \mathbf{x}')$ and $\boldsymbol{\Omega}(\mathbf{x}, \mathbf{x}') = \boldsymbol{\Upsilon}(\mathbf{x}, \mathbf{x}')\boldsymbol{\Upsilon}^\top(\mathbf{x}, \mathbf{x}') = \boldsymbol{\Gamma}\mathbf{Z}(\mathbf{x}, \mathbf{x}')\mathbf{Z}^\top(\mathbf{x}, \mathbf{x}')\boldsymbol{\Gamma}^\top \in \mathbf{R}^{D\times D}$ follows a spatial Wishart process $\mathcal{SW}(P, \boldsymbol{\Gamma}\boldsymbol{\Gamma}^\top, \{\rho_d(\mathbf{x}, \mathbf{x}')\}_{d=1}^D)$. In [34], authors assume $\boldsymbol{\Gamma} = \mathbf{I}_D$ and $\rho_d(\mathbf{x}, \mathbf{x}')$ is the same for all values of d. Inference in this model is accomplished using Markov chain Monte Carlo. For details about the particular inference procedure, the reader is referred to [34].

5.3 Process Convolutions

More general non-separable kernels can also be constructed from a generative point of view. We saw in section 4.2.1 that the linear model of coregionalization involves instantaneous mixing through a linear weighted sum of independent processes to construct correlated processes. By instantaneous mixing we mean that the output function $\mathbf{f}(\mathbf{x})$ evaluated at the input point $\mathbf{x}$ only depends on the values of the latent functions $\{u_q(\mathbf{x})\}_{q=1}^{Q}$ at the same input $\mathbf{x}$. This instantaneous mixing leads to a kernel function for vector-valued functions that has a separable form.

A non-trivial way to mix the latent functions is through convolving a base process with a smoothing kernel.[6] If the base process is a Gaussian process, it turns out that the convolved process is also a Gaussian process. We can therefore exploit convolutions to construct covariance functions [8, 102, 43, 44, 14, 56, 2].

In a similar way to the linear model of coregionalization, we consider Q groups of functions, where a particular group q has elements $u_q^i(\mathbf{z})$, for $i = 1, \ldots, R_q$. Each member of the group has the same covariance $k_q(\mathbf{x}, \mathbf{x}')$, but is sampled independently. Any output $f_d(\mathbf{x})$ is described by

$$f_d(\mathbf{x}) = \sum_{q=1}^{Q} \sum_{i=1}^{R_q} \int_{\mathcal{X}} G_{d,q}^i(\mathbf{x} - \mathbf{z}) u_q^i(\mathbf{z}) \mathrm{d}\mathbf{z} + w_d(\mathbf{x}) = \sum_{q=1}^{Q} f_d^q(\mathbf{x}) + w_d(\mathbf{x}),$$

where

$$f_d^q(\mathbf{x}) = \sum_{i=1}^{R_q} \int_{\mathcal{X}} G_{d,q}^i(\mathbf{x} - \mathbf{z}) u_q^i(\mathbf{z}) \mathrm{d}\mathbf{z}, \tag{27}$$

and $\{w_d(\mathbf{x})\}_{d=1}^{D}$ are independent Gaussian processes with zero mean and covariance $k_{w_d}(\mathbf{x}, \mathbf{x}')$. For the integrals in equation (27) to exist, it is assumed that each kernel $G_{d,q}^i(\mathbf{x})$ is a continuous function with compact support [46] or square-integrable [102, 44]. The kernel $G_{d,q}^i(\mathbf{x})$ is also known as the moving average function [102] or the smoothing kernel [44]. We have included the superscript q for $f_d^q(\mathbf{x})$ in (27) to emphasize the fact that the function depends on the set of latent processes $\{u_q^i(\mathbf{x})\}_{i=1}^{R_q}$. The latent functions $u_q^i(\mathbf{z})$ are Gaussian processes with general covariances $k_q(\mathbf{x}, \mathbf{x}')$.

Under the same independence assumptions used in the linear model of coregionalization, the covariance between $f_d(\mathbf{x})$ and $f_{d'}(\mathbf{x}')$ follows

$$(\mathbf{K}(\mathbf{x}, \mathbf{x}'))_{d,d'} = \sum_{q=1}^{Q} k_{f_d^q, f_{d'}^q}(\mathbf{x}, \mathbf{x}') + k_{w_d}(\mathbf{x}, \mathbf{x}') \delta_{d,d'}, \tag{28}$$

where

$$k_{f_d^q, f_{d'}^q}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{R_q} \int_{\mathcal{X}} G_{d,q}^i(\mathbf{x} - \mathbf{z}) \int_{\mathcal{X}} G_{d',q}^i(\mathbf{x}' - \mathbf{z}') k_q(\mathbf{z}, \mathbf{z}') \mathrm{d}\mathbf{z}' \mathrm{d}\mathbf{z}. \tag{29}$$

Specifying $G_{d,q}^i(\mathbf{x} - \mathbf{z})$ and $k_q(\mathbf{z}, \mathbf{z}')$ in the equation above, the covariance for the outputs $f_d(\mathbf{x})$ can be constructed indirectly. Notice that if the smoothing kernels are taken to be the Dirac delta function in equation (29), such that $G_{d,q}^i(\mathbf{x} - \mathbf{z}) = a_{d,q}^i \delta(\mathbf{x} - \mathbf{z})$,[7] the double integral is easily solved and the linear model of coregionalization is recovered. In this respect, process convolutions could also be seen as a dynamic version of the linear model of coregionalization in the sense that the latent functions are dynamically transformed with the help of the kernel smoothing functions, as opposed to a static mapping of the latent functions in the LMC case. See section 5.3.1 for a comparison between the process convolution and the LMC.

A recent review of several extensions of this approach for the single output case is presented in [17]. Some of those extensions include the construction of nonstationary covariances [43, 45, 30, 31, 73] and spatiotemporal covariances [109, 107, 108].

The idea of using convolutions for constructing multiple output covariances was originally proposed by [102]. They assumed that $Q = 1$, $R_q = 1$, that the process $u(\mathbf{x})$ was white Gaussian noise and that the input space was

[6] We use kernel to refer to both reproducing kernels and smoothing kernels. Smoothing kernels are functions which are convolved with a signal to create a smoothed version of that signal.

[7] We have slightly abused of the delta notation to indicate the Kronecker delta for discrete arguments and the Dirac function for continuous arguments. The particular meaning should be understood from the context.

$\mathcal{X} = \mathbf{R}^p$. [44] depicted a similar construction to the one introduced by [102], but partitioned the input space into disjoint subsets $\mathcal{X} = \bigcap_{d=0}^{D} \mathcal{X}_d$, allowing dependence between the outputs only in certain subsets of the input space where the latent process was common to all convolutions.[8]

Higdon [44] coined the general moving average construction to develop a covariance function in equation (27) as a *process convolution*.

Boyle and Frean [14, 15] introduced the process convolution approach for multiple outputs to the machine learning community with the name of "dependent Gaussian processes" (DGP), further developed in [13]. They allow the number of latent functions to be greater than one ($Q \geq 1$). In [56] and [2], the latent processes $\{u_q(\mathbf{x})\}_{q=1}^{Q}$ followed a more general Gaussian process that goes beyond the white noise assumption.

5.3.1 Comparison Between Process Convolutions and LMC

Figure 6 shows an example of the instantaneous mixing effect obtained in the ICM and the LMC, and the non-instantaneous mixing effect due to the process convolution framework. We sampled twice from a two-output Gaussian process with an ICM covariance with $R_q = 1$ (first column), an LMC covariance with $R_q = 2$ (second column) and a process convolution covariance with $R_q = 1$ and $Q = 1$ (third column). As in the examples for the LMC, we use EQ kernels for the basic kernels $k_q(\mathbf{x}, \mathbf{x}')$. We also use an exponentiated quadraticform for the smoothing kernel functions $G_{1,1}^1(\mathbf{x} - \mathbf{x}')$ and $G_{2,1}^1(\mathbf{x} - \mathbf{x}')$ and assume that the latent function is white Gaussian noise.

Notice from Figure 6 that samples from the ICM share the same length-scale. Samples from the LMC are weighted sums of functions with different length-scales (a long length-scale term and a short length-scale term). In both models, ICM and LMC, the two outputs share the same length-scale or the same combination of length-scales. Samples from the PC show that the contribution from the latent function is different over each output. Output $f_1(x)$ has a long length-scale behavior, while output $f_2(x)$ has a short length-scale behavior.

It would be possible to get similar samples to the PC ones using a LMC. We would need to assume, though, that some covariances in a particular coregionalization matrix $\mathbf{B}_q$ are zero. In Figure 7, we display samples from a LMC with $R_q = 2$ and $Q = 2$. We have forced $b_{1,1}^2 = b_{1,2}^2 = b_{2,1}^2 = 0$. To generate these samples we use an ICM with $R_q = 2$ and a latent function with long length-scale, and then add a sample from an independent Gaussian process with a short length-scale to output $f_2(x)$. It is debatable if this compound model (ICM plus independent GP) would capture the relevant correlations between the output functions.

To summarize, the choice of a kernel corresponds to specifying dependencies among inputs and outputs. In the linear model of co-regionalization this is done considering separately inputs, via the kernels k_q, and outputs, via the coregionalization matrices $\mathbf{B}_q$, for $q = 1, \ldots, Q$. Having a large large value of Q allows for a larger expressive power of the model. For example if the output components are substantially different functions (different smoothness or length scale), we might be able to capture their variability by choosing a sufficiently large Q. This is at the expense of a larger computational burden.

On the other hand, the process convolution framework attempts to model the variance of the set of outputs by the direct association of a different smoothing kernel $G_d(\mathbf{x})$ to each output $f_d(\mathbf{x})$. By specifying $G_d(\mathbf{x})$, one can model, for example, the degree of smoothness and the length-scale that characterizes each output. If each output happens to have more than one degree of variation (marginally, it is a sum of functions of varied smoothness) one is faced with the same situation as in LMC, namely, the need to augment the parameter space so as to satisfy a required precision. However, due to the local description of each output that the process convolution performs, it is likely that the parameter space for the process convolution approach grows slower than the parameter space for LMC.

5.3.2 Other Approaches Related to Process Convolutions

In [61], a different moving average construction for the covariance of multiple outputs was introduced. It is obtained as a convolution over covariance functions in contrast to the process convolution approach where the convolution is performed over processes. Assuming that the covariances involved are isotropic and the only latent function $u(\mathbf{x})$ is a white Gaussian noise, [61] show that the cross-covariance obtained from

$$\operatorname{cov}\left[f_d(\mathbf{x}+\mathbf{h}), f_{d'}(\mathbf{x})\right] = \int_{\mathcal{X}} k_d(\mathbf{h}-\mathbf{z}) k_{d'}(\mathbf{z}) \mathrm{d}\mathbf{z},$$

[8] The latent process $u(\mathbf{x})$ was assumed to be independent on these separate subspaces.

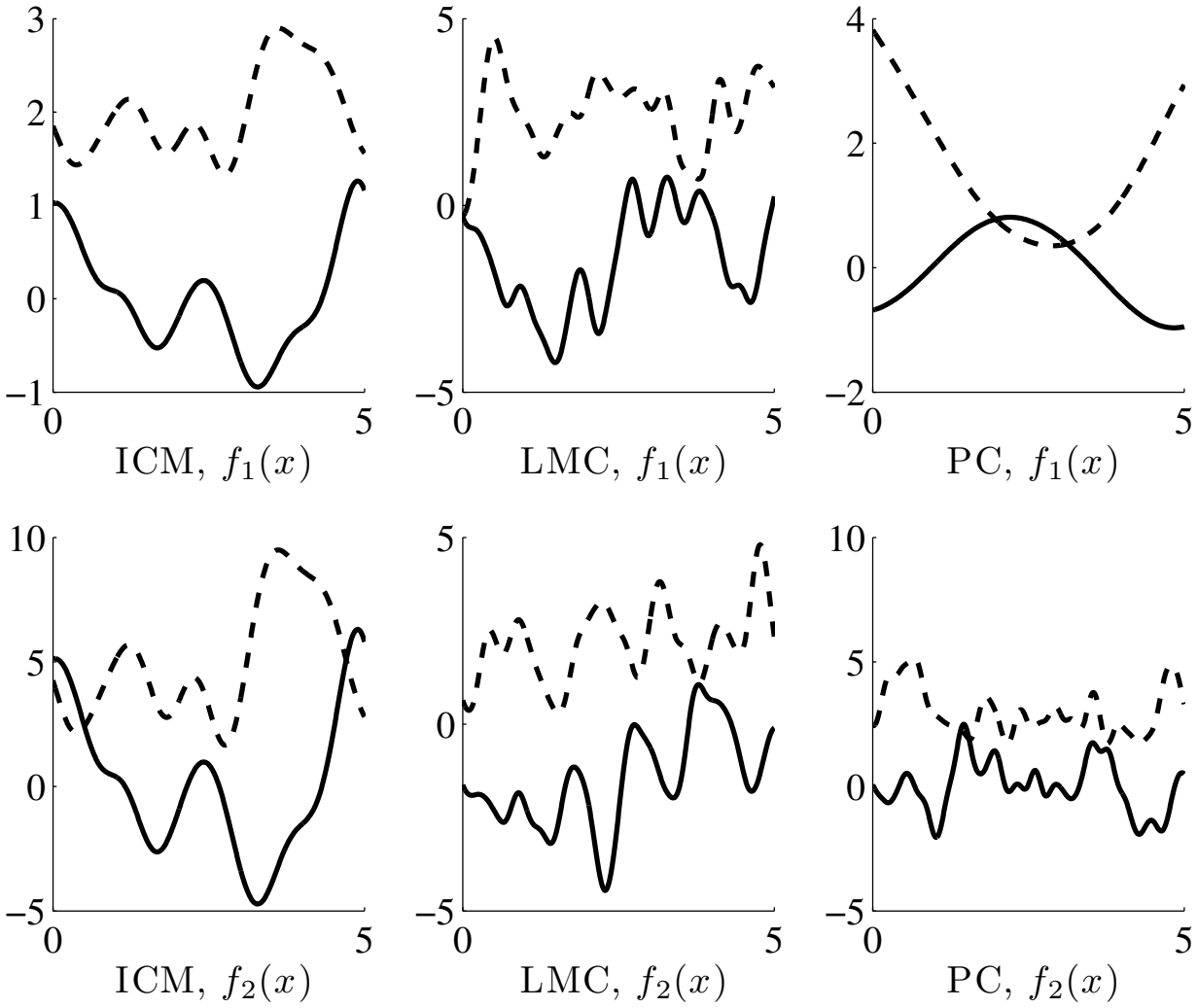

Figure 6: Two samples from three kernel matrices obtained using the intrinsic coregionalization model with $R_q = 1$ (first column), the linear model of coregionalization with $R_q = 2$ (second column) and the process convolution formalism with $R_q = 1$ and $Q = 1$ (third column). Solid lines represent one of the samples. Dashed lines represent the other sample. There are two outputs, one row per output. Notice that for the ICM and the LMC, the outputs have the same length-scale (in the ICM case) or combined length-scales (in the LMC case). The outputs generated from the process convolution covariance differ in their relative length-scale.

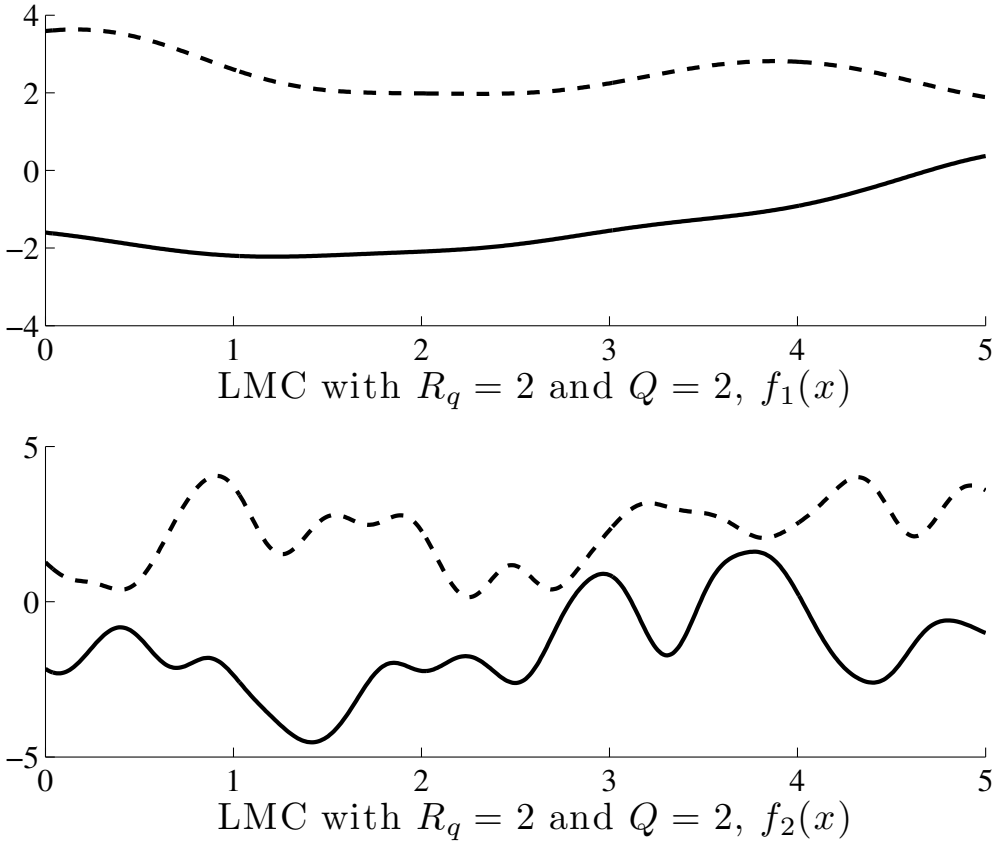

Figure 7: Two samples from a linear model of coregionalization with $R_q = 2$ and $Q = 2$. The solid lines represent one of the samples. The dashed lines represent the other sample. Samples share a long length-scale behavior. An added short length-scale term appears only in output two.

where $k_d(\mathbf{h})$ and $k_{d'}(\mathbf{h})$ are covariances associated to the outputs d and d', lead to a valid covariance function for the outputs $\{f_d(\mathbf{x})\}_{d=1}^{D}$. If we assume that the smoothing kernels are not only square integrable, but also positive definite functions, then the covariance convolution approach turns out to be a particular case of the process convolution approach (square-integrability might be easier to satisfy than positive definiteness).

[67] introduced the idea of transforming a Gaussian process prior using a discretized process convolution, $\mathbf{f}_d = \mathbf{G}_d\mathbf{u}$, where $\mathbf{G}_d \in \mathbf{R}^{N\times M}$ is a so called *design matrix* with elements $\{\mathbf{G}_d(\mathbf{x}_n, \mathbf{z}_m)\}_{n=1,m=1}^{N,M}$ and $\mathbf{u}^\top = [u(\mathbf{x}_1), \ldots, u(\mathbf{x}_M)]$. Such a transformation could be applied for the purposes of fusing the information from multiple sensors, for solving inverse problems in reconstruction of images or for reducing computational complexity working with the filtered data in the transformed space [92]. Convolutions with general Gaussian processes for modelling single outputs, were also proposed by [30, 31], but instead of the continuous convolution, [30, 31] used a discrete convolution. The purpose in [30, 31] was to develop a spatially varying covariance for single outputs, by allowing the parameters of the covariance of a base process to change as a function of the input domain.

Process convolutions are closely related to the Bayesian kernel method [77, 58] construct reproducible kernel Hilbert spaces (RKHS) by assigning priors to signed measures and mapping these measures through integral operators. In particular, define the following space of functions,

$$\mathcal{F} = \Big\{ f \Big| f(x) = \int_{\mathcal{X}} G(x, z)\gamma(\mathrm{d}z),\ \gamma \in \Gamma \Big\},$$

for some space $\Gamma \subseteq \mathcal{B}(\mathcal{X})$ of signed Borel measures. In [77, proposition 1], the authors show that for $\Gamma = \mathcal{B}(\mathcal{X})$, the space of all signed Borel measures, $\mathcal{F}$ corresponds to a RKHS. Examples of these measures that appear in the

form of stochastic processes include Gaussian processes, Dirichlet processes and Lévy processes. In principle, we can extend this framework for the multiple output case, expressing each output as

$$f_d(x) = \int_{\mathcal{X}} G_d(x, z)\gamma(\mathrm{d}z).$$

6 Inference and Computational Considerations

Practical use of multiple-output kernel functions require the tuning of the hyperparameters, and dealing with the issue of computational burden related directly with the inversion of matrices of dimension $ND \times ND$. Cross-validation and maximization of the log-marginal likelihood are alternatives for parameter tuning, while matrix diagonalization and reduced-rank approximations are choices for overcoming computational complexity of the matrix inversion.

In this section we refer to the parameter estimation problem for the models presented in section 4 and 5 and also to the computational complexity when using those models in practice.

6.1 Estimation of Parameters in Regularization Theory

From a regularization perspective, once the kernel is fixed, to find a solution we need to solve the linear system defined in (5). The regularization parameter as well as the possible kernel parameters are typically tuned via cross-validation. The kernel free-parameters are usually reduced to one or two scalars (e.g. the width of a scalar kernel). While considering for example separable kernels the matrix $\mathbf{B}$ is fixed by design, rather than learned, and the only free parameters are those of the scalar kernel.

Solving problem (5), this is $\overline{\mathbf{c}} = (\mathbf{K}(\mathbf{X}, \mathbf{X}) + \lambda N \mathbf{I})^{-1}\overline{\mathbf{y}}$, is in general a costly operation both in terms of memory and time. When we have to solve the problem for a single value of λ Cholesky decomposition is the method of choice, while when we want to compute the solution for different values of λ (for example to perform cross validation) singular valued decomposition (SVD) is the method of choice. In both case the complexity in the worst case is $O(D^3N^3)$ (with a larger constant for the SVD) and the associated storage requirement is $O(D^2N^2)$

As observed in [7], this computational burden can be greatly reduced for separable kernels. For example, if we consider the kernel $\mathbf{K}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}')\mathbf{I}$ the kernel matrix $\mathbf{K}(\mathbf{X}, \mathbf{X})$ becomes block diagonal. In particular if the input points are the same, all the blocks are equal and the problem reduces to inverting an N by N matrix. The simple example above serves as a prototype for the more general case of a kernel of the form $\mathbf{K}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}')\mathbf{B}$. The point is that for this class of kernels, we can use the eigen-system of the matrix $\mathbf{B}$ to define a new coordinate system where the kernel matrix becomes block diagonal.

We start observing that if we denote with $(\sigma_1, \mathbf{u}_1), \ldots, (\sigma_D, \mathbf{u}_D)$ the eigenvalues and eigenvectors of $\mathbf{B}$ we can write the matrix $\mathbf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_N)$, with $\mathbf{c}_i \in \mathbf{R}^D$, as $\mathbf{C} = \sum_{d=1}^{D} \tilde{\mathbf{c}}^d \otimes \mathbf{u}_d$, where $\tilde{\mathbf{c}}^d = (\langle \mathbf{c}_1, \mathbf{u}_d \rangle_D, \ldots, \langle \mathbf{c}_N, \mathbf{u}_d \rangle_D)$ and $\otimes$ is the tensor product and similarly $\tilde{\mathbf{Y}} = \sum_{d=1}^{D} \tilde{\mathbf{y}}^d \otimes \mathbf{u}_d$, with $\tilde{\mathbf{y}}^d = (\langle \mathbf{y}_1, \mathbf{u}_d \rangle_D, \ldots, \langle \mathbf{y}_N, \mathbf{u}_d \rangle_D)$. The above transformations are simply rotations in the output space. Moreover, for the considered class of kernels, the kernel matrix $\mathbf{K}(\mathbf{X}, \mathbf{X})$ is given by the tensor product of the $N \times N$ scalar kernel matrix $k(\mathbf{X}, \mathbf{X})$ and $\mathbf{B}$, that is $\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X})$.

Then we have the following equalities

$$\begin{aligned}\mathbf{C} = (\mathbf{K}(\mathbf{X}, \mathbf{X}) + N\lambda_N \mathbf{I})^{-1}\mathbf{Y} &= \sum_{d=1}^{D} (\mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}) + N\lambda_N \mathbf{I})^{-1} \tilde{\mathbf{y}}^d \otimes \mathbf{u}_d \\ &= \sum_{d=1}^{D} (\sigma_d k(\mathbf{X}, \mathbf{X}) + N\lambda_N \mathbf{I})^{-1} \tilde{\mathbf{y}}^d \otimes \mathbf{u}_d.\end{aligned}$$

Since the eigenvectors $\mathbf{u}_j$ are orthonormal, it follows that:

$$\tilde{\mathbf{c}}^d = (\sigma_d k(\mathbf{X}, \mathbf{X}) + N\lambda_N \mathbf{I})^{-1} \tilde{\mathbf{y}}^j = \left(k(\mathbf{X}, \mathbf{X}) + \frac{\lambda_N}{\sigma_d}\mathbf{I}\right)^{-1} \frac{\tilde{\mathbf{y}}^d}{\sigma_d}, \tag{30}$$

for $d = 1, \ldots, D$. The above equation shows that in the new coordinate system we have to solve D essentially independent problems after rescaling each kernel matrix by σ_d or equivalently rescaling the regularization parameter (and the outputs). The above calculation shows that all kernels of this form allow for a simple implementation at the price of the eigen-decomposition of the matrix $\mathbf{B}$. Then we see that the computational cost is now essentially

$O(D^3) + O(N^3)$ as opposed to $O(D^3N^3)$ in the general case. Also, it shows that the coupling among the different tasks can be seen as a rotation and rescaling of the output points. Stegle *et al.* [94] also applied this approach in the context of fitting matrix variate Gaussian models with spherical noise.

6.2 Parameters Estimation for Gaussian Processes

In machine learning parameter estimation for Gaussian processes is often approached through maximization of the marginal likelihood. The method also goes by the names of evidence approximation, type II maximum likelihood, empirical Bayes, among others [10].

With a Gaussian likelihood and after integrating $\mathbf{f}$ using the Gaussian prior, the marginal likelihood is given by

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\phi}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}(\mathbf{X}, \mathbf{X}) + \boldsymbol{\Sigma}), \tag{31}$$

where $\boldsymbol{\phi}$ are the hyperparameters.

The objective function is the logarithm of the marginal likelihood

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\phi}) = -\frac{1}{2}\mathbf{y}^\top (\mathbf{K}(\mathbf{X}, \mathbf{X}) + \boldsymbol{\Sigma})^{-1}\mathbf{y} - \frac{1}{2}\log |\mathbf{K}(\mathbf{X}, \mathbf{X}) + \boldsymbol{\Sigma}| - \frac{ND}{2}\log 2\pi. \tag{32}$$

The parameters $\boldsymbol{\phi}$ are obtained by maximizing $\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\phi})$ with respect to each element in $\boldsymbol{\phi}$. Maximization is performed using a numerical optimization algorithm, for example, a gradient based method. Derivatives follow

$$\frac{\partial \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\phi})}{\partial \phi_i} = \frac{1}{2}\mathbf{y}^\top \overline{\mathbf{K}(\mathbf{X}, \mathbf{X})}^{-1} \frac{\partial \overline{\mathbf{K}(\mathbf{X}, \mathbf{X})}}{\partial \phi_i} \overline{\mathbf{K}(\mathbf{X}, \mathbf{X})}^{-1}\mathbf{y} - \frac{1}{2}\operatorname{trace}\left(\overline{\mathbf{K}(\mathbf{X}, \mathbf{X})}^{-1} \frac{\partial \overline{\mathbf{K}(\mathbf{X}, \mathbf{X})}}{\partial \phi_i}\right), \tag{33}$$

where ϕ_i is an element of the vector $\boldsymbol{\phi}$ and $\overline{\mathbf{K}(\mathbf{X}, \mathbf{X})} = \mathbf{K}(\mathbf{X}, \mathbf{X}) + \boldsymbol{\Sigma}$. In the case of the LMC, in which the coregionalization matrices must be positive semidefinite, it is possible to use an incomplete Cholesky decomposition $\mathbf{B}_q = \widetilde{\mathbf{L}}_q \widetilde{\mathbf{L}}_q^\top$, with $\widetilde{\mathbf{L}}_q \in \mathbf{R}^{D \times R_q}$, as suggested in [12]. The elements of the matrices $\mathbf{L}_q$ are considered part of the vector $\boldsymbol{\phi}$.

Another method used for parameter estimation, more common in the geostatistics literature, consists of optimizing an objective function which involves some empirical measure of the correlation between the functions $f_d(\mathbf{x})$, $\widehat{\mathbf{K}}(\mathbf{x}, \mathbf{x}')$, and the multivariate covariance obtained using a particular model, $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ [38, 53, 76]. Assuming stationary covariances, this criteria reduces to

$$\text{WSS} = \sum_{i=1}^{N} w(\mathbf{h}_i) \operatorname{trace}\left\{\left[\left(\widehat{\mathbf{K}}(\mathbf{h}_i) - \mathbf{K}(\mathbf{h}_i)\right)\right]^2\right\}, \tag{34}$$

where $\mathbf{h}_i = \mathbf{x}_i - \mathbf{x}_i'$ is a lag vector, $w(\mathbf{h}_i)$ is a weight coefficient, $\widehat{\mathbf{K}}(\mathbf{h}_i)$ is an experimental covariance matrix with entries obtained by different estimators for cross-covariance functions [75, 102], and $\mathbf{K}(\mathbf{h}_i)$ is the covariance matrix obtained, for example, using the linear model of coregionalization.[9] One of the first algorithms for estimating the parameter vector $\boldsymbol{\phi}$ in LMC was proposed by [38]. It assumed that the parameters of the basic covariance functions $k_q(\mathbf{x}, \mathbf{x}')$ had been determined a priori and then used a weighted least squares method to fit the coregionalization matrices. In [76] the efficiency of other least squares procedures was evaluated experimentally, including ordinary least squares and generalized least squares. Other more general algorithms in which all the parameters are estimated simultaneously include simulated annealing [54] and the EM algorithm [114]. Ver Hoef and Barry [102] also proposed the use of an objective function like (34), to estimate the parameters in the covariance obtained from a process convolution.

[9] Note that the common practice in geostatistics is to work with variograms instead of covariances. A variogram characterizes a general class of random functions known as intrinsic random functions [62], which are random processes whose increments follow a stationary second-order process. For clarity of exposition, we will avoid the introduction of the variogram and its properties. The interested reader can follow the original paper by [62] for a motivation of their existence, [36] for a comparison between variograms and covariance functions and [37] for a definition of the linear model of coregionalization in terms of variograms.

Both methods described above, the evidence approximation or the least-square method, give point estimates of the parameter vector $\boldsymbol{\phi}$. Several authors have employed full Bayesian inference by assigning priors to $\boldsymbol{\phi}$ and computing the posterior distribution through some sampling procedure. Examples include [42] and [23] under the LMC framework or [14] and [99] under the process convolution approach.

As mentioned before, for non-Gaussian likelihoods, there is not a closed form solution for the posterior distribution nor for the marginal likelihood. However, the marginal likelihood can be approximated under a Laplace, variational Bayes or expectation propagation (EP) approximation frameworks for multiple output classification [93, 11], and used to find estimates for the hyperparameters. Hence, the error function is replaced for $\log q(\mathbf{y}|\mathbf{X}, \boldsymbol{\phi})$, where $q(\mathbf{y}|\mathbf{X}, \boldsymbol{\phi})$ is the approximated marginal likelihood. Parameters are again estimated using a gradient based methods.

The problem of computational complexity for Gaussian processes in the multiple output context has been studied by different authors [83, 103, 96, 13, 2, 1]. Fundamentally, the computational problem is the same than the one appearing in regularization theory, that is, the inversion of the matrix $\overline{\mathbf{K}(\mathbf{X}, \mathbf{X})} = \mathbf{K}(\mathbf{X}, \mathbf{X}) + \boldsymbol{\Sigma}$ for solving equation (5). This step is necessary for computing the marginal likelihood and its derivatives (for estimating the hyperparameters as explained before) or for computing the predictive distribution. With the exception of the method by [83], the approximation methods proposed in [103, 96, 13, 2, 1] can be applied to reduce computational complexity, whichever covariance function (LMC or process convolution, for example) is used to compute the multi-output covariance matrix. In other words, the computational efficiency gained is independent of the particular method employed to compute the covariance matrix.

Before looking with some detail at the different approximation methods employed in the Gaussian processes literature for multiple outputs, it is worth mentioning that computing the kernel function through process convolutions in equation (29) implies solving a double integral, which is not always feasible for any choice of the smoothing kernels $G_{d,q}^{i}(\cdot)$ and covariance functions $k_q(\mathbf{x}, \mathbf{x}')$. An example of an analytically tractable covariance function occurs when both the smoothing kernel and the covariance function for the latent functions have EQ kernels [2], or when the smoothing kernels have an exponentiated quadratic form and the latent functions are Gaussian white noise processes [73, 14]. An alternative would be to consider discrete process convolutions [44] instead of the continuous process convolution of equations (28) and (29), avoiding in this way the need to solve double integrals.

We now briefly summarize different methods for reducing computational complexity in multi-output Gaussian processes.

As we mentioned before, Rougier [83] assumes that the multiple output problem can be seen as a single output problem considering the output index as another variable of the input space. The predicted output, $f(\mathbf{x}_*)$ is expressed as a weighted sum of Q deterministic regressors that explain the mean of the output process plus a Gaussian error term that explains the variance in the output. Both, the set of regressors and the covariance for the error are assumed to be separable in the input space. The covariance takes the form $k(\mathbf{x}, \mathbf{x}')k_T(d, d')$, as in the introduction of section 4. For isotopic models ([83] refers to this condition as regular outputs, meaning outputs that are evaluated at the same set of inputs $\mathbf{X}$), the mean and covariance for the output, can be obtained through Kronecker products for the regressors and the covariances involved in the error term. For inference the inversion of the necessary terms is accomplished using properties of the Kronecker product. For example, if $\mathbf{K}(\mathbf{X}, \mathbf{X}') = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}')$, then $\mathbf{K}^{-1}(\mathbf{X}, \mathbf{X}') = \mathbf{B}^{-1} \otimes k^{-1}(\mathbf{X}, \mathbf{X}')$. Computational complexity is reduced to $O(D^3) + O(N^3)$, similar to the eigendecomposition method in section 6.1.

Ver Hoef and Barry [103] present a simulation example with $D = 2$. Prediction over one of the variables is performed using *cokriging*. In cokriging scenarios, usually one has access to a few measurements of a primary variable, but plenty of observations for a secondary variable. In geostatistics, for example, predicting the concentration of heavy pollutant metals (say Cadmium or Lead), which are expensive to measure, can be done using inexpensive and oversampled variables as a proxy (say pH levels) [37]. Following a suggestion by [95] (page 172), the authors partition the secondary observations into subgroups of observations and assume the likelihood function is the sum of the partial likelihood functions of several systems that include the primary observations and each of the subgroups of the secondary observations. In other words, the joint probability distribution $p(f_1(\mathbf{X}_1), f_2(\mathbf{X}_2))$ is factorised as $p(f_1(\mathbf{X}_1), f_2(\mathbf{X}_2)) = \prod_{j=1}^{J} p(f_1(\mathbf{X}_1), f_2^{(j)}(\mathbf{X}_2^{(j)}))$, where $f_2^{(j)}(\mathbf{X}_2^{(j)})$ indicates the observations in the subgroup j out of J subgroups of observations, for the secondary variable. Inversion of the particular covariance matrix derived from these assumptions grows as $O(JN^3)$, where N is the number of input points per secondary variable.

Also, the authors use a fast Fourier transform for computing the autocovariance matrices $(\mathbf{K}(\mathbf{X}_d, \mathbf{X}_d))_{d,d}$ and cross-covariance matrices $(\mathbf{K}(\mathbf{X}_d, \mathbf{X}_{d'}))_{d,d'}$.

Boyle [13] proposed an extension of the *reduced rank approximation* method presented by [80], to be applied

to the dependent Gaussian process construction. The author outlined the generalization of the methodology for $D = 2$. The outputs $f_1(\mathbf{X}_1)$ and $f_2(\mathbf{X}_2)$ are defined as

$$\begin{bmatrix} f_1(\mathbf{X}_1) \\ f_2(\mathbf{X}_2) \end{bmatrix} = \begin{bmatrix} (\mathbf{K}(\mathbf{X}_1, \mathbf{X}_1))_{1,1} & (\mathbf{K}(\mathbf{X}_1, \mathbf{X}_2))_{1,2} \\ (\mathbf{K}(\mathbf{X}_2, \mathbf{X}_1))_{2,1} & (\mathbf{K}(\mathbf{X}_2, \mathbf{X}_2))_{2,2} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{w}}_1 \\ \widetilde{\mathbf{w}}_2 \end{bmatrix},$$

where $\widetilde{\mathbf{w}}_d$ are vectors of weights associated to each output including additional weights corresponding to the test inputs, one for each output. Based on this likelihood, a predictive distribution for the joint prediction of $f_1(\mathbf{X})$ and $f_2(\mathbf{X})$ can be obtained, with the characteristic that the variance for the approximation, approaches to the variance of the full predictive distribution of the Gaussian process, even for test points away from the training data. The elements in the matrices $(\mathbf{K}(\mathbf{X}_d, \mathbf{X}_{d'}))_{d,d'}$ are computed using the covariances and cross-covariances developed in sections 4 and 5. Computational complexity reduces to $O(DNM^2)$, where N is the number of sample points per output and M is an user specified value that accounts for the rank of the approximation.

In [2], the authors show how through making specific conditional independence assumptions, inspired by the model structure in the process convolution formulation (for which the LMC is a special case), it is possible to arrive at a series of efficient approximations that represent the covariance matrix $\mathbf{K}(\mathbf{X}, \mathbf{X})$ using a reduced rank approximation $\mathbf{Q}$ plus a matrix $\mathbf{D}$, where $\mathbf{D}$ has a specific structure that depends on the particular independence assumption made to obtain the approximation. Approximations can reduce the computational complexity to $O(NDM^2)$ with M representing a user specified value that determines the rank of $\mathbf{Q}$. Approximations obtained in this way, have similarities with the conditional approximations summarized for a single output in [81].

Finally, the informative vector machine (IVM) [57] has also been extended to Gaussian processes using kernel matrices derived from particular versions of the linear model of coregionalization, including [96] and [55]. In the IVM, only a smaller subset of size M of the data points is chosen for constructing the GP predictor. The data points selected are the ones that maximize a differential entropy score [55] or an information gain criteria [96]. Computational complexity for this approximation is again $O(NDM^2)$.

For the computational complexities shown above, we assumed $R_q = 1$ and $Q = 1$.

7 Applications of Multivariate Kernels

In this chapter we further describe in more detail some of the applications of kernel approaches to multi-output learning from the statistics and machine learning communities.

One of the main application areas of multivariate Gaussian process has been in computer emulation. In [29], the LMC is used as the covariance function for a Gaussian process emulator of a finite-element method that solves for frequency response functions obtained from a structure. The outputs correspond to pairs of masses and stiffnesses for several structural modes of vibration for an aircraft model. The input space is made of variables related to physical properties, such as Tail tip mass or Wingtip mass, among others.

Multivariate computer emulators are also frequently used for modelling time series. We mentioned this type of application in section 4.2.4. Mostly, the number of time points in the time series are matched to the number of outputs (we expressed this as $D = T$ before), and different time series correspond to different input values for the emulation. The particular input values employed are obtained from different ranges that the input variables can take (given by an expert), and are chosen according to some space-filling criteria (Latin hypercube design, for example) [84]. In [23], the time series correspond to the evolution of the net biome productivity (NBP) index, which in turn is the output of the Sheffield dynamic global vegetation model. In [63], the time series is the temperature of a particular location of a container with decomposing foam. The simulation model is a finite element model and simulates the transfer of heat through decomposing foam.

In machine learning the range of applications for multivariate kernels is increasing. In [72], the ICM is used to model the dependencies of multivariate time series in a sensor network. Sensors located in the south coast of England measure different environmental variables such as temperature, wind speed, tide height, among others. Sensors located close to each other make similar readings. If there are faulty sensors, their missing readings could be interpolated using the healthy ones.

In [22], the authors use the ICM for obtaining the inverse dynamics of a robotic manipulator. The inverse dynamics problem consists in computing the torques at different joints of the robotic arm, as function of the angle, angle velocity and angle acceleration for the different joints. Computed torques are necessary to drive the robotic arm along a particular trajectory. Furthermore, the authors consider several *contexts*, this is, different dynamics due to different loadings at the end effector. Joints are modelled independently using an ICM for each of them, being the outputs the different contexts and being the inputs, the angles, the angle velocities and the angle accelerations. Besides interpolation, the model is also used for extrapolation of novel contexts.

The authors of [11] use the ICM for preference elicitation, where a user is prompted to solve simple queries in order to receive a recommendation. The ICM is used as a covariance function for a GP that captures dependencies between users (through the matrix $\mathbf{B}$), and dependencies between items (through the covariance $k(\mathbf{x}, \mathbf{x}')$).

In [56] and [33], the authors use a process convolution to model the interaction between several genes and a transcription factor protein, in a gene regulatory network. Each output corresponds to a gene, and each latent function corresponds to a transcription factor protein. It is assummed that transcription factors regulate the rate at which particular genes produce primary RNA. The output functions and the latent functions are indexed by time. The smoothing kernel functions $G^i_{d,q}(\cdot)$ correspond to the impulse response obtained from an ordinary differential equation of first order. Given gene expression data, the problem is to infer the time evolution of the transcription factor.

In [3], the authors use a process convolution to model the dependencies between different body parts of an actor that performs modern dancing movements. This type of data is usually known as mocap (for motion capture) data. The outputs correspond to time courses of angles referenced to a root node, for each body part modelled. The smoothing kernel used corresponds to a Green's function arising from a second order ordinary differential equation.

In [67], the authors use a discretized process convolution for solving an inverse problem in reconstruction of images, and for fusing the information from multiple sensors.

In [16], two particulate matter (PM) levels measured in the air (10 μm in diameter and 2.5 μm in diameter), at different spatial locations, are modeled as the added influence of coarse and fine particles. In turn, these coarse and fine particles are modeled as random walks and then transformed by discrete convolutions to represent the levels of PM at 10 μm and 2.5 μm. The objective is to extract information about PM at 2.5 μm from the abundant readings of PM at 10 μm.

8 Discussion

We have presented a survey of multiple output kernel functions to be used in kernel methods including regularization theory and Gaussian processes. From the regularization theory point of view, the multiple output problem can be seen as a regularization method that minimizes directly a loss function while constraining the parameters of the learned vector-valued function. In a Gaussian process framework, from a machine learning context, the multiple output problem is equivalent to formulate a generative model for each output that expresses correlations as functions of the output function index and the input space, using a set of common latent functions.

We presented two general families of kernels for vector-valued functions including the separable family (including the SoS kernels) and different instantiations of what we would call the nonseparable family. The separable family represent the kernel function as the product of a kernel for the outputs, independently of the value that the input can have, and a kernel function for the input space, independently of the output index. The most general model is the linear model of coregionalization, with many other kernel functions that appear in the machine learning literature as particular cases. Within the family of nonseparable kernels, the process convolution construction has proved useful for several theoretical and applied problems and as we have seen before, it can be considered as a generalization of the linear model of coregionalization.

Model selection establishes a path for future research in multiple-output kernels related problems. From a Bayesian perspective, in the setup of LMC and process convolutions, model selection includes principled mechanisms to find the number of latent functions and/or the rank of the coregionalization matrices. More general model selection problems involve the ability to test if given some data, the outputs are really correlated or influence each other, compared to the simpler assumption of independence. Other model selection problem includes the influence of the input space configuration (isotopic against heterotopic) towards the sharing of strengths between outputs. Although such problems have been studied to some extent in the geostatistics literature, there remain open issues.

Acknowledgements

MA and NL are very grateful for support from a Google Research Award "Mechanistically Inspired Convolution Processes for Learning" and the EPSRC Grant No EP/F005687/1 "Gaussian Processes for Systems Identification with Applications in Systems Biology". MA also acknowledges the support from the Overseas Research Student Award Scheme (ORSAS), from the School of Computer Science of the University of Manchester and from the Universidad Tecnológica de Pereira, Colombia. LR would like to thank Luca Baldassarre and Tomaso Poggio for

many useful discussions. LR is assistant professor at DISI, University of Genova and currently on leave of absence. We also thank to two anonymous reviewers for their helpful comments.

This work was supported in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. Thanks to PASCAL 2 support the authors of this paper organized two workshops: *Statistics and Machine Learning Interface Meeting* (see `http://intranet.cs.man.ac.uk/mlo/slim09/`), across 23-24 of July, 2009 at Manchester, UK and *Kernels for Multiple Outputs and Multi-task Learning: Frequentist and Bayesian Points of View* (see `http://intranet.cs.man.ac.uk/mlo/mock09/`) held on December 12 at Whistler, Canada as part as one of the Workshops of NIPS 2009. This publication only reflects the authors' views, but they benefited greatly by interactions with other researchers at these workshops who included Andreas Argyriou, David Higdon, Tom Fricker, Sayan Mukherjee, Tony O'Hagan, Ian Vernon, Hans Wackernagel, Richard Wilkinson, and Chris Williams.

Notation

Generalities

p dimensionality of the input space
D number of outputs
N, N_d number of data points for output d
Q number of latent functions (for generative models)
$\mathcal{X}$ input space
$\mathbf{X}_d$ input training data for output d, $\mathbf{X}_d = \{\mathbf{x}_{d,n}\}_{n=1}^{N_d}$
$\mathbf{X}$ input training data for all outputs, $\mathbf{X} = \{\mathbf{X}_d\}_{d=1}^{D}$

Functions

$k(\mathbf{x}, \mathbf{x}')$ general scalar kernel
$\mathbf{K}(\mathbf{x}, \mathbf{x}')$ general kernel valued matrix with entries $(\mathbf{K}(\mathbf{x}, \mathbf{x}'))_{d,d'}$ with $d, d = 1, \ldots, D$
$k_q(\mathbf{x}, \mathbf{x}')$ scalar kernel for the $q-$th latent function
$f_d(\mathbf{x})$ d-th output evaluated at $\mathbf{x}$
$\mathbf{f}(\mathbf{x})$, vector-valued function, $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_D(\mathbf{x})]^\top$
$\delta_{k,k'}$ Kronecker delta for discrete arguments
$\delta(x)$ Dirac delta for continuous arguments

Vectors and matrices

$k_q(\mathbf{X}, \mathbf{X})$ kernel matrix with entries $k_q(\mathbf{x}, \mathbf{x}')$ evaluated at $\mathbf{X}$
$\mathbf{f}_d(\mathbf{X}_d)$ $\mathbf{f}_d(\mathbf{x})$ evaluated at $\mathbf{X}_d$, $\mathbf{f}_d = [f_d(\mathbf{x}_{d,1}), \ldots, f_d(\mathbf{x}_{d,N_d})]^\top$
$\mathbf{f}(\mathbf{X})$ vectors $\{\mathbf{f}_d\}_{d=1}^{D}$, stacked in a column vector
$(\mathbf{K}(\mathbf{X}_d, \mathbf{X}_{d'}))_{d,d'}$ kernel matrix with entries $(\mathbf{K}(\mathbf{x}_{d,n}, \mathbf{x}_{d',m}))_{d,d'}$ with $\mathbf{x}_{d,n} \in \mathbf{X}_d$ and $\mathbf{x}_{d',m} \in \mathbf{X}_{d'}$
$\mathbf{K}(\mathbf{X}, \mathbf{X})$ kernel matrix with blocks $(\mathbf{K}(\mathbf{X}_d, \mathbf{X}_{d'}))_{d,d'}$ with $d, d' = 1, \ldots, D$
$\mathbf{I}_N$ identity matrix of size N

References

[1] Mauricio A. Álvarez. *Convolved Gaussian Process Priors for Multivariate Regression with Applications to Dynamical Systems*. PhD thesis, School of Computer Science, University of Manchester, Manchester, UK, 2011.

[2] Mauricio A. Álvarez and Neil D. Lawrence. Sparse convolved Gaussian processes for multi-output regression. In Koller et al. [52], pages 57–64.

[3] Mauricio A. Álvarez, David Luengo, and Neil D. Lawrence. Latent Force Models. In David van Dyk and Max Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, pages 9–16, Clearwater Beach, Florida, 16-18 April 2009. JMLR W&CP 5.

[4] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[5] Andreas Argyriou, Andreas Maurer, and Massimiliano Pontil. An algorithm for transfer learning in a heterogeneous environment. In *ECML/PKDD (1)*, pages 71–85, 2008.

[6] N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.

[7] L. Baldassarre, L. Rosasco, A. Barla, and A. Verri. Multi-output learning via spectral filtering. Technical report, Massachusetts Institute of Technology, 2011. MIT-CSAIL-TR-2011-004, CBCL-296.

[8] Ronald Paul Barry and Jay M. Ver Hoef. Blackbox kriging: spatial prediction without specifying variogram models. *Journal of Agricultural, Biological and Environmental Statistics*, 1(3):297–322, 1996.

[9] M. J. Bayarri, James O. Berger, Marc C. Kennedy, Athanasios Kottas, Rui Paulo, Jerry Sacks, John A. Cafeo, Chin-Hsu Lin, and Jian Tu. Predicting vehicle crashworthiness: Validation of computer models for functional and hierarchical data. *Journal of the American Statistical Association*, 104(487):929–943, 2009.

[10] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[11] Edwin Bonilla, Shengbo Guo, and Scott Sanner. Gaussian process preference elicitation. In *NIPS*, volume 24, pages 262–270, Cambridge, MA, 2011. MIT Press.

[12] Edwin V. Bonilla, Kian Ming Chai, and Christopher K. I. Williams. Multi-task Gaussian process prediction. In John C. Platt, Daphne Koller, Yoram Singer, and Sam Roweis, editors, *NIPS*, volume 20, Cambridge, MA, 2008. MIT Press.

[13] Phillip Boyle. *Gaussian Processes for Regression and Optimisation*. PhD thesis, Victoria University of Wellington, Wellington, New Zealand, 2007.

[14] Phillip Boyle and Marcus Frean. Dependent Gaussian processes. In Saul et al. [85], pages 217–224.

[15] Phillip Boyle and Marcus Frean. Multiple output Gaussian process regression. Technical Report CS-TR-05/2, School of Mathematical and Computing Sciences, Victoria University, New Zealand, 2005.

[16] Catherine A. Calder. A dynamic process convolution approach to modeling ambient particulate matter concentrations. *Environmetrics*, 19:39–48, 2008.

[17] Catherine A. Calder and Noel Cressie. Some topics in convolution-based spatial modeling. In *Proceedings of the 56th Session of the International Statistics Institute*, August 2007.

[18] A. Caponnetto, C.A. Micchelli, M. Pontil, and Y. Ying. Universal kernels for multi-task learning. *Journal of Machine Learning Research*, 9:1615–1646, 2008.

[19] C. Carmeli, E. De Vito, and A. Toigo. Vector valued reproducing kernel Hilbert spaces of integrable functions and Mercer theorem. *Anal. Appl. (Singap.)*, 4(4):377–408, 2006.

[20] Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[21] Kian Ming Chai. Generalization errors and learning curves for regression with multi-task Gaussian processes. In Yoshua Bengio, Dale Schuurmans, John Laferty, Chris Williams, and Aron Culotta, editors, *NIPS*, volume 22, pages 279–287, Cambridge, MA, 2010. MIT Press.

[22] Kian Ming A. Chai, Christopher K. I. Williams, Stefan Klanke, and Sethu Vijayakumar. Multi-task Gaussian process learning of robot inverse dynamics. In Koller et al. [52], pages 265–272.

[23] Stefano Conti and Anthony O'Hagan. Bayesian emulation of complex multi-output and dynamic computer models. *Journal of Statistical Planning and Inference*, 140(3):640–651, 2010.

[24] Noel A. C. Cressie. *Statistics for Spatial Data*. John Wiley & Sons (Revised edition), USA, 1993.

[25] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bull. Amer. Math. Soc. (N.S.)*, 39(1):1–49 (electronic), 2002.

[26] E. De Vito, L. Rosasco, A. Caponnetto, M. Piana, and A. Verri. Some properties of regularized kernel methods. *Journal of Machine Learning Research*, 5:1363–1390, 2004.

[27] T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

[28] Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

[29] Thomas E. Fricker, Jeremy E. Oakley, Neil D. Sims, and Keith Worden. Probabilistic uncertainty analysis of an frf of a structure using a Gaussian process emulator. *Mechanical Systems and Signal Processing*, 25(8):2962–2975, 2011.

[30] Montserrat Fuentes. Interpolation of nonstationary air pollution processes: a spatial spectral approach. *Statistical Modelling*, 2:281–298, 2002.

[31] Montserrat Fuentes. Spectral methods for nonstationary spatial processes. *Biometrika*, 89(1):197–210, 2002.

[32] E.J. Fuselier Jr. *Refined error estimates for matrix-valued radial basis functions*. PhD thesis, Texas A&M University, 2006.

[33] Pei Gao, Antti Honkela, Magnus Rattray, and Neil D. Lawrence. Gaussian process modelling of latent chemical species: Applications to inferring transcription factor activities. *Bioinformatics*, 24:i70–i75, 2008.

[34] Alan E. Gelfand, Alexandra M. Schmidt, Sudipto Banerjee, and C.F. Sirmans. Nonstationary multivariate process modeling through spatially varying coregionalization. *TEST*, 13(2):263–312, 2004.

[35] F. Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63:169–176, 1989.

[36] Tilmann Gneiting, Zoltán Sasvári, and Martin Schlather. Analogies and correspondences between variograms and covariance functions. *Advances in Applied Probability*, 33(3):617–630, 2001.

[37] Pierre Goovaerts. *Geostatistics For Natural Resources Evaluation*. Oxford University Press, USA, 1997.

[38] Michel Goulard and Marc Voltz. Linear coregionalization model: Tools for estimation and choice of cross-variogram matrix. *Mathematical Geology*, 24(3):269–286, 1992.

[39] J.A. Vargas Guzmán, A.W. Warrick, and D.E. Myers. Coregionalization by linear combination of nonorthogonal components. *Mathematical Geology*, 34(4):405–419, 2002.

[40] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, second edition, 2009.

[41] Jeffrey D. Helterbrand and Noel Cressie. Universal cokriging under intrinsic coregionalization. *Mathematical Geology*, 26(2):205–226, 1994.

[42] Dave Higdon, Jim Gattiker, Brian Williams, and Maria Rightley. Computer model calibration using high dimensional output. *Journal of the American Statistical Association*, 103(482):570–583, 2008.

[43] David M. Higdon. A process-convolution approach to modeling temperatures in the north atlantic ocean. *Journal of Ecological and Environmental Statistics*, 5:173–190, 1998.

[44] David M. Higdon. Space and space-time modelling using process convolutions. In C. Anderson, V. Barnett, P. Chatwin, and A. El-Shaarawi, editors, *Quantitative methods for current environmental issues*, pages 37–56. Springer-Verlag, 2002.

[45] David M. Higdon, Jenise Swall, and John Kern. Non-stationary spatial modeling. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics 6*, pages 761–768. Oxford University Press, 1998.

[46] Lars Hörmander. *The analysis of Linear Partial Differential Operators I*. Springer-Verlag, Berlin Hiedelberg, first edition, 1983.

[47] L. Jacob, F. Bach, and J.P. Vert. Clustered multi-task learning: A convex formulation. In *Advances in Neural Information Processing Systems (NIPS)*. Curran Associates, Inc, 2008.

[48] Laurent Jacob, Francis Bach, and Jean-Philippe Vert. Clustered multi-task learning: A convex formulation. In *NIPS 21*, pages 745–752, 2008.

[49] Andre G. Journel and Charles J. Huijbregts. *Mining Geostatistics*. Academic Press, London, 1978.

[50] Hideto Kazawa, Tomonori Izumitani, Hirotoshi Taira, and Eisaku Maeda. Maximal margin labeling for multi-topic text categorization. In Saul et al. [85], pages 649–656.

[51] G. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation of stochastic processes and smoothing by splines. *Ann. Math. Stat.*, 41:495–502, 1970.

[52] Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors. *NIPS*, volume 21, Cambridge, MA, 2009. MIT Press.

[53] H.R. Künsch, A. Papritz, and F. Bassi. Generalized cross-covariances and their estimation. *Mathematical Geology*, 29(6):779–799, 1997.

[54] R.M. Lark and A. Papritz. Fitting a linear model of coregionalization for soil properties using simulated annealing. *Geoderma*, 115:245–260, 2003.

[55] Neil D. Lawrence and John C. Platt. Learning to learn with the informative vector machine. In *Proceedings of the 21st International Conference on Machine Learning (ICML 2004)*, pages 512–519, 2004.

[56] Neil D. Lawrence, Guido Sanguinetti, and Magnus Rattray. Modelling transcriptional regulation using Gaussian processes. In Bernhard Schölkopf, John C. Platt, and Thomas Hofmann, editors, *NIPS*, volume 19, pages 785–792, Cambridge, MA, 2007. MIT Press.

[57] Neil D. Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In Sue Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, volume 15, pages 625–632, Cambridge, MA, 2003. MIT Press.

[58] Feng Liang, Kai Mao, Ming Liao, Sayan Mukherjee, and Mike West. Non-parametric Bayesian kernel models. Department of Statistical Science, Duke University, Discussion Paper 07-10. (Submitted for publication), 2009.

[59] S. Lowitzsch. A density theorem for matrix-valued radial basis functions. *Numerical Algorithms*, 39(1):253–256, 2005.

[60] I. Macêdo and R. Castro. Learning divergence-free and curl-free vector fields with matrix-valued kernels. Technical report, Instituto Nacional de Matematica Pura e Aplicada, 2008.

[61] Anandamayee Majumdar and Alan E. Gelfand. Multivariate spatial modeling for geostatistical data using convolved covariance functions. *Mathematical Geology*, 39(2):225–244, 2007.

[62] Georges Matheron. The intrinsic random functions and their applications. *Advances in Applied Probability*, 5(3):439–468, 1973.

[63] John McFarland, Sankaran Mahadevan, Vicente Romero, and Laura Swiler. Calibration and Uncertainty Analysis for Computer Simulations with Multivariate Output. *AIAA Journal*, 46(5):1253–1265, 2008.

[64] C. A. Micchelli and M. Pontil. Kernels for multi-task learning. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2004.

[65] C.A. Micchelli and M. Pontil. On learning vector–valued functions. *Neural Computation*, 17:177–204, 2005.

[66] Thomas P. Minka and Rosalind W. Picard. Learning how to learn is learning with point sets, 1999. Revised version 1999 available at `http://research.microsoft.com/en-us/um/people/minka/papers/point-sets.html`.

[67] Roderick Murray-Smith and Barak A. Pearlmutter. Transformation of Gaussian process priors. In Joab Winkler, Mahesan Niranjan, and Neil Lawrence, editors, *Deterministic and Statistical Methods in Machine Learning*, pages 110–123. LNAI 3635, Springer-Verlag, 2005.

[68] F.J. Narcowich and J.D. Ward. Generalized hermite interpolation via matrix-valued conditionally positive definite functions. *Mathematics of Computation*, 63(208):661–687, 1994.

[69] G. Obozinski, B. Taskar, and M.I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2):231–252, 2010.

[70] Anthony O'Hagan. Bayesian analysis of computer code outputs: A tutorial. *Reliability Engineering and System Safety*, 91:1290–1300, 2006.

[71] Michael A. Osborne and Stephen J. Roberts. Gaussian processes for prediction. Technical report, Department of Engineering Science, University of Oxford, 2007.

[72] Michael A. Osborne, Alex Rogers, Sarvapali D. Ramchurn, Stephen J. Roberts, and Nicholas R. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN 2008)*, 2008.

[73] Christopher J. Paciorek and Mark J. Schervish. Nonstationary covariance functions for Gaussian process regression. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[74] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345 –1359, oct. 2010.

[75] A. Papritz, H.R. Künsch, and R. Webster. On the pseudo cross-variogram. *Mathematical Geology*, 25(8):1015–1026, 1993.

[76] Bernard Pelletier, Pierre Dutilleul, Guillaume Larocque, and James W. Fyles. Fitting the linear model of coregionalization by generalized least squares. *Mathematical Geology*, 36(3):323–343, 2004.

[77] Natesh S. Pillai, Qiang Wu, Feng Liang, Sayan Mukherjee, and Robert L. Wolpert. Characterizing the function space for Bayesian kernel models. *Journal of Machine Learning Research*, 8:1769–1797, 2007.

[78] Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[79] Peter Z. G Qian, Huaiqing Wu, and C. F. Jeff Wu. Gaussian process models for computer experiments with qualitative and quantitative factors. *Technometrics*, 50(3):383–396, 2008.

[80] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. Analysis of some methods for reduced rank Gaussian process regression. In R. Murray-Smith and R. Shorten, editors, *Lecture Notes in Computer Science*, volume 3355, pages 98–127. Springer, 2005.

[81] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

[82] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[83] Jonathan Rougier. Efficient emulators for multivariate deterministic functions. *Journal of Computational and Graphical Statistics*, 17(4):827–834, 2008.

[84] Thomas J. Santner, Brian J. Williams, and William I. Notz. *The Design and Analysis of Computer Experiments*. Springer, first edition, 2003.

[85] Lawrence Saul, Yair Weiss, and Léon Bouttou, editors. *NIPS*, volume 17, Cambridge, MA, 2005. MIT Press.

[86] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. The MIT Press, USA, 2002.

[87] L. Schwartz. Sous-espaces hilbertiens d'espaces vectoriels topologiques et noyaux associés (noyaux reproduisants). *J. Analyse Math.*, 13:115–256, 1964.

[88] Bernhard Schšlkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *In Proceedings of the Annual Conference on Computational Learning Theory*, pages 416–426, 2001.

[89] Matthias Seeger and Michael I. Jordan. Sparse Gaussian Process Classification With Multiple Classes. Technical Report 661, Department of Statistics, University of California at Berkeley, 2004.

[90] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[91] D. Sheldon. Graphical multi-task learning. Technical report, Cornell University, 2008. Preprint.

[92] Jian Qing Shi, Roderick Murray-Smith, D.M. Titterington, and Barak Pearlmutter. Learning with large data sets using filtered Gaussian process priors. In R. Murray-Smith and R. Shorten, editors, *Proceedings of the Hamilton Summer School on Switching and Learning in Feedback systems*, pages 128–139. LNCS 3355, Springer-Verlag, 2005.

[93] Grigorios Skolidis and Guido Sanguinetti. Bayesian multitask classification with Gaussian process priors. *IEEE Transactions on Neural Networks*, 22(12):2011 – 2021, 2011.

[94] Oliver Stegle, Christoph Lippert, Joris Mooij, Neil Lawrence, and Karsten Borgwardt. Learning sparse inverse covariance matrices in the presence of confounders. In *Neural Information Processing Systems*, 2011.

[95] Michael L. Stein. *Interpolation of Spatial Data*. Springer-Verlag New York, Inc., first edition, 1999.

[96] Yee Whye Teh, Matthias Seeger, and Michael I. Jordan. Semiparametric latent factor models. In Robert G. Cowell and Zoubin Ghahramani, editors, *AISTATS 10*, pages 333–340, Barbados, 6-8 January 2005. Society for Artificial Intelligence and Statistics.

[97] Sebastian Thrun. Is learning the n-thing any easier than learning the first? In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *NIPS*, volume 08, pages 640–646, Cambridge, MA, 1996. MIT Press.

[98] A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill Posed Problems*. W. H. Winston, Washington, D.C., 1977.

[99] Michalis Titsias, Neil D Lawrence, and Magnus Rattray. Efficient sampling for Gaussian process inference using control variables. In Koller et al. [52], pages 1681–1688.

[100] V. N. Vapnik. *Statistical learning theory*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. John Wiley & Sons Inc., New York, 1998. A Wiley-Interscience Publication.

[101] E. Vazquez and E. Walter. Multi-output support vector regression. *13th IFAC Symposium on System Identification*, 2003.

[102] Jay M. Ver Hoef and Ronald Paul Barry. Constructing and fitting models for cokriging and multivariable spatial prediction. *Journal of Statistical Plannig and Inference*, 69:275–294, 1998.

[103] Jay M. Ver Hoef, Noel Cressie, and Ronald Paul Barry. Flexible spatial models for kriging and cokriging using moving averages and the Fast Fourier Transform (FFT). *Journal of Computational and Graphical Statistics*, 13(2):265–282, 2004.

[104] Hans Wackernagel. *Multivariate Geostatistics*. Springer-Verlag Heidelberg New york, 2003.

[105] Grace Wahba. *Spline Models for Observational Data*. SIAM, first edition, 1990.

[106] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, 2008.

[107] Christopher K. Wikle. A kernel-based spectral model for non-Gaussian spatio-temporal processes. *Statistical Modelling*, 2:299–314, 2002.

[108] Christopher K. Wikle. Hierarchical Bayesian models for predicting the spread of ecological processes. *Ecology*, 84(6):1382–1394, 2003.

[109] Christopher K. Wikle, L. Mark Berliner, and Noel Cressie. Hierarchical Bayesian space-time models. *Environmental and Ecological Statistics*, 5:117–154, 1998.

[110] Christopher K.I. Williams and David Barber. Bayesian Classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

[111] Ian Woodward, Mark R. Lomas, and Richard A. Betts. Vegetation-climate feedbacks in a greenhouse world. *Philosophical Transactions: Biological Sciences*, 353(1365):29–39, 1998.

[112] Ya Xue, Xuejun Liao, and Lawrence Carin. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

[113] Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning Gaussian processes from multiple tasks. In *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, pages 1012–1019, 2005.

[114] Hao Zhang. Maximum-likelihood estimation for multivariate spatial linear coregionalization models. *Environmetrics*, 18:125–139, 2007.