Building a Bayes Linear Emulator

This document complements the code provided in the GitHub repository https://github.com/dario-domi/Bayes-Linear-Emulation. It illustrates the components which a Bayes Linear Emulator of a function f comprises of, and details the meaning of all optional arguments of the R function `BL.Emul()` in the repository: the latter is available to download, for the user to build their own emulator in research tasks.

The repository also contains the script `Emulation_Example.R`, which provides two examples of use of `BL.Emul()` in a toy yet highly non-linear 2D problem.

Setting

Let $f: U \to \mathbb{R}$ be a function, $U \subseteq \mathbb{R}^m$. Suppose f is slow to evaluate[1]. We thus want to build a fast surrogate of f, which we call an **emulator** of f. The emulator will provide instantaneous predictions of the value $f(x)$ for any $x \in U$, and determine the level of uncertainty associated with the prediction.

The process behind the construction of an emulator consists of two main steps:

1. Make prior specifications about f (*e.g.*, mean behaviour, smoothness, *etc*).
2. Adjust these in light of the observed values of f at a small, fixed set of inputs $\mathcal{D} = \{x^{(1)}, \dots, x^{(n)}\} \subset U$.

The emulator built by the R function `BL.Emul()` relies upon the Bayes Linear statistical approach[2]: informally, this means that only second-order specifications (mean, variances and covariances) about f are required, which are then *adjusted* once the values $f(x^{(1)}), \dots, f(x^{(n)})$ are observed.

[1] Typically, $f(x)$ is the output of a computer model, simulating the dynamics of a complex system when values of the system parameters have been set to x. Evaluating $f(x)$ may be both slow and computationally intensive.

[2] See: M. Goldstein and D. Wooff. *Bayes Linear Statistics: Theory and Methods*. Wiley, 2006.

1 Model of the BL Emulator

In order to build an emulator $\widetilde{f}$ of f, we assume the following form for f:

$$f(x) \;=\; \sum_{j=1}^{q} \beta_j\, g_j(x_{[A]}) \;+\; \eta(x_{[A]}) \;+\; \varepsilon(x), \qquad x \in U\,. \tag{1}$$

$x_{[A]} \in \mathbb{R}^p$ are called **active inputs**: they are identified as a subset of $p \leq m$ coordinates of $x \in \mathbb{R}^m$, responsible for most of the variation of f across the domain U.

The three terms in (1) are as follows:

1. The **regression term** $\boldsymbol{\beta}^T \boldsymbol{g}(x_{[A]})$ is a linear combination of q known basis functions $g_j(x_{[A]})$, *e.g.*, polynomials in the components of $x_{[A]}$. $\boldsymbol{\beta} \in \mathbb{R}^q$ are unknown coefficients, for which second-order prior specification can be made.

2. The **term** $\eta(x_{[A]})$ is a zero-mean stochastic process, with prior covariance function modelled as:

$$\mathrm{Cov}\big[\eta(x_{[A]}),\, \eta(x'_{[A]})\big] = \sigma_\eta^2\, k\Big(\big\|x_{[A]} - x'_{[A]}\big\|_{\boldsymbol{d}}\Big). \tag{2}$$

   The **kernel** $k(\cdot)$ is a function of a scalar quantity $r \geq 0$, in (2) this being a norm of the difference between the two active inputs $x_{[A]}$ and $x'_{[A]}$. The norm depends on a set of **correlation lengths** $\boldsymbol{d} = (d_1, \ldots, d_p)$.

   The role of the correlation lengths and classical choices for k available within `BL.Emul()` are discussed in subsection 1.1. The kernel k satisfies $k(0) = 1$, hence the quantity $\sigma_\eta^2 \geq 0$ represents the variance of $\eta(x_{[A]})$ at any $x_{[A]}$.

3. The **nugget term** $\varepsilon(x)$ accounts for the small portion of variability in f explained by the inactive inputs (residual variability). It is modelled as a zero-mean stochastic process, with $\mathrm{Var}[\varepsilon(x)] = \sigma_\varepsilon^2$ and $\mathrm{Cov}[\varepsilon(x), \varepsilon(x')] = 0$ if $x \neq x'$.

We denote the sum of the first two terms in (1) by $f^*(x_{[A]})$, so that equation (1) reads

$$f(x) \;=\; f^*(x_{[A]}) \;+\; \varepsilon(x)\,. \tag{3}$$

Given the above specifications and the values of f at the set $\mathcal{D} = \{x^{(1)}, \ldots, x^{(n)}\}$ of **design points**:

$$F = \big(f(x^{(1)}), \ldots, f(x^{(n)})\big)^T,$$

the Bayes Linear framework allows to *adjust* the prior model for $f^*(x_{[A]})$ to the observed values in F, at any $x_{[A]}$.

This means that an emulator $\widetilde{f}^*$ of f^* can be built, which provides, at any $x_{[A]}$, the adjusted mean and variance of $f^*(x_{[A]})$ in light of the observed F:

$$\mathbb{E}\big[\widetilde{f}^*(x_{[A]})\big], \quad \mathbb{V}\mathrm{ar}\big[\widetilde{f}^*(x_{[A]})\big]. \tag{4}$$

The interested reader can find the formulas and derivation of the expressions in (4) in Section***. Once these are obtained, expectation and variance of an emulator of f (in which we are ultimately interested), at an input x with active inputs $x_{[A]}$, follow from the relation (3) linking f and f^*:

$$\mathbb{E}\big[\widetilde{f}(x)\big] = \mathbb{E}\big[\widetilde{f}^*(x_{[A]})\big] \tag{5}$$

$$\mathbb{V}\mathrm{ar}\big[\widetilde{f}(x)\big] = \mathbb{V}\mathrm{ar}\big[\widetilde{f}^*(x_{[A]})\big] + \sigma_\varepsilon^2\ . \tag{6}$$

The function `BL.Emul()` returns the values in (5) and (6) at any sequence of N inputs $x_{[A]}$ passed to the function.

Computational Note: The function is optimised for speed, as well as memory: it will run smoothly even for very large N, as long as there is enough RAM space to handle the $2N$ floats returned as outputs.

1.1 Covariance Kernel and Correlation Lengths

This section provides a guide to the kernels $k(\cdot)$ available in `BL.Emul()` (see equation (2)), simple heuristics to help choose between them, and the role of correlation lengths in specifying the prior covariance of η.

Given a vector $\boldsymbol{d} = (d_1, \ldots, d_p)$ with $d_i > 0$, the square of the norm in (2) is computed as follows:

$$\big\|x_{[A]} - x'_{[A]}\big\|_{\boldsymbol{d}}^2 = \sum_{i=1}^{p} \left(\frac{x_{[A],i} - x'_{[A],i}}{d_i}\right)^2, \qquad x_{[A]}, x'_{[A]} \in \mathbb{R}^p, \tag{7}$$

where $x_{[A],i}$ denotes the i-th component of $x_{[A]} \in \mathbb{R}^p$. The effect of d_i is therefore to scale distances along coordinate i: the smaller d_i, the “more distant” the two vectors $x_{[A]}$ and $x'_{[A]}$ will appear along coordinate i.

Once $r = \|x_{[A]} - x'_{[A]}\|_{\boldsymbol{d}}$ is evaluated, the prior correlation between $\eta(x_{[A]})$ and $\eta(x'_{[A]})$ is computed as $k(r)$ where k is one of the kernels described below. In all cases, k is a decreasing function of $r \geq 0$ with $k(0) = 1$. Hence, higher values of the correlation lengths in $\boldsymbol{d}$ yield higher prior correlations between the various $\eta(x_{[A]})$.

While the values of $\boldsymbol{d}$ affect the strength of correlations in the outputs of η, the specific choice of the kernel k will affect the regularity of η, and may therefore be made in light of prior knowledge/beliefs about the regularity of the function f to be emulated. The following choices for k are available in `BL.Emul()`:

- **Squared-Exponential**

$$k(r) = \exp\left(-r^2\right), \quad r \geq 0 \,. \tag{8}$$

  A process η with the above covariance kernel is infinitely many times differentiable.

- **Matérn 5/2**

$$k(r) = \left(1 + \sqrt{5}r + \frac{5}{3}r^2\right) \exp\left(-\sqrt{5}r\right), \quad r \geq 0. \tag{9}$$

  A process η with the above covariance kernel is twice differentiable.

- **Matérn 3/2**

$$k(r) = \left(1 + \sqrt{3}r\right) \exp\left(-\sqrt{3}r\right), \quad r \geq 0. \tag{10}$$

  A process η with the above covariance kernel is differentiable.

- **Absolute Exponential**

$$k(r) = \exp\left(-r\right), \quad r \geq 0. \tag{11}$$

  A process η with the above covariance kernel is continuous but not differentiable.

All previous four regularity results hold both pathwise and in mean-square.

2 Table of Inputs of `BL.Emul()`

Table 1 lists all arguments of the function `BL.Emul()`, their size/type, and the meaning of each argument in reference to the above setting. Default values of all optional arguments are also listed.

Table 1: Description of all inputs to the function `BL.Emul()`. Notation as in section 1. Design points: $x^{(i)}$, $i = 1, \dots, n$. Test points (where to evaluate the emulator): $\widetilde{x}^{(i)}$, $i = 1, \dots, N$. The default value for correlation length d_j is computed as a third of the maximum difference between the j-th components of any two design points.

Variable (`X`)	Size	Description	Optional?	Default
`ActInp.Design`	$n \times p$	`X[i,j]`$= x^{(i)}_{[A],j}$	✗	—
`ActInp.Test`	$N \times p$	`X[i,j]`$= \widetilde{x}^{(i)}_{[A],j}$	✗	—
`y`	n	`y[i]`$= f(x^{(i)})$	✗	—
`Regress.Design`	$n \times q$	`X[i,j]`$= g_j(x^{(i)}_{[A]})$	✓	`ActInp.Design`
`Regress.Test`	$N \times q$	`X[i,j]`$= g_j(\widetilde{x}^{(i)}_{[A]})$	✓	`ActInp.Test`
`beta`	q	Prior expect of $\boldsymbol{\beta}$: `beta[j]`$= \mathbb{E}[\beta_j]$	✓	q-dim zero vector
`Cov.beta`	$q \times q$	Prior cov of $\boldsymbol{\beta}$: `Cov.beta`$= \mathbb{C}\mathrm{ov}[\boldsymbol{\beta}]$	✓	$q \times q$ zero matrix
`sigma2`	scalar	σ^2_η (eq. (2))	✓	`var(y)`
`kernel`	string	One of: `exp2`, `abs_exp`, `matern32`, `matern52`	✓	`exp2`
`d`	p	Corr lengths: `d[j]`$= d_j$	✓	see caption
`nu`	scalar	σ^2_ε (var of $\varepsilon(x)$ in (1))	✓	0