To-do list

1. Can't I just standardise x? . . . 3
2. Show ridge in the log-likelihood plot. . . . 13
3. Attempt to prove this. . . . 19

Contents

Haziq Jamil
Department of Statistics
London School of Economics and Political Science
PhD thesis: 'Regression modelling using Fisher information covariance kernels (I-priors)'

Chapter 4

Regression modelling using I-priors

In the previous chapter, we defined an I-prior for the normal regression model (1.1) subject to (1.2) and f belonging to a reproducing kernel Hilbert or Krein space of functions $\mathcal{F}$, as a Gaussian distribution on f with covariance function equal to the Fisher information for f. We also saw how new function spaces can be constructed via the polynomial and ANOVA RKKS. In this chapter, we shall describe various regression models, and identify them with appropriate RKKSs, so that an I-prior may be defined on it.

Methods for estimating I-prior models are described in Section 4.2. Estimation here refers to obtaining the posterior distribution of the regression function under an I-prior, while optimising the kernel parameters of $\mathcal{F}$ and the error precision $\boldsymbol{\Psi}$. Likelihood based methods, namely direct optimisation of the likelihood and the expectation-maximisation (EM) algorithm, are the preferred estimation methods of choice. Having said this, it is also possible to estimate I-prior models under a full Bayesian paradigm by employing Markov chain Monte Carlo methods to sample from the relevant posterior densities.

Careful considerations of the computational aspects are required to ensure efficient estimation of I-prior models, and these are discussed in Section 4.3. The culmination of the computational work on I-prior estimation is the **iprior** package (Jamil and Bergsma, 2017), which is a publicly available R package that has been published to CRAN.

Finally, several examples of I-prior modelling are presented in Section 4.5: in particular, a multilevel data set, a longitudinal data set, and a data set involving a functional covariate, are analysed using the I-prior methodology.

sec:various-regression

4.1 Various regression models

In the introductory chapter (Section 1.1), we described several interesting regression models. The goal of this section is to formulate the I-prior model that describes each of these models. This is done by carefully choosing the RKHS/RKKS $\mathcal{F}$ of real functions over a set $\mathcal{X}$ to which the regression function f belongs. Without loss of generality and for simplicity, assume a prior mean of zero for the I-prior distribution.

4.1.1 Multiple linear regression

Let $\mathcal{X} \equiv \mathbb{R}^p$ be equipped with the regular Euclidean dot product, and $\mathcal{F}_\lambda$ be the scaled canonical RKHS of functions over $\mathcal{X}$ with kernel $h_\lambda(\mathbf{x}, \mathbf{x}') = \lambda \mathbf{x}^\top \mathbf{x}'$, for any two $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^p$. Then, an I-prior on f implies that

$$\begin{aligned} f(\mathbf{x}_i) &= \sum_{j=1}^{n} \lambda \mathbf{x}_i^\top \mathbf{x}_j w_j \\ &= \sum_{j=1}^{n} \lambda \left(\sum_{k=1}^{p} x_{ik} x_{jk} \right) w_j \\ &= \beta_1 x_{i1} + \cdots + \beta_p x_{ip}, \end{aligned}$$

where each $\beta_k := \lambda \sum_{j=1}^{n} x_{jk} w_j$. This implies a multivariate normal prior distribution for the regression coefficients

$$\boldsymbol{\beta} := (\beta_1, \ldots, \beta_p) \sim \mathrm{N}_p(\mathbf{0}, \lambda^2 \mathbf{X}^\top \boldsymbol{\Psi} \mathbf{X}), \tag{4.1}$$

eq:ipriorcanonical

where $\mathbf{X}$ is the $n \times p$ design matrix for the covariates, excluding the column of ones at the beginning typically reserved for the intercept. As expected, the covariance matrix for $\boldsymbol{\beta}$ is recognised as the scaled Fisher information matrix for the regression coefficients.

If the covariates are not scaled similarly, then the values of f are incoherent—if x_1 measures weight in kilograms and x_2 height in centimetres, what measurement does $\beta_1 x_1 + \beta_2 x_2$ represent? To overcome this, one could decompose the regression function into

1. Can't I just standardise x?

$$f(\mathbf{x}_i) = f_1(x_{i1}) + \cdots + f_p(x_{ip})$$

for which $f \in \mathcal{F}_\lambda \equiv \mathcal{F}_{\lambda_1} \oplus \cdots \oplus \mathcal{F}_{\lambda_p}$, and $\mathcal{F}_{\lambda_k}$, $k = 1, \ldots, p$ are unidimensional canonical RKHSs with kernels $h_{\lambda_k}(x_{ik}, x_{jk}) = \lambda_k x_{ik} x_{jk}$. In effect, we now have p scale parameters,

one for each of the RKKSs associated with the p covariates. The RKKS $\mathcal{F}_\lambda$ therefore has kernel

$$h(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{p} \lambda_k x_{ik} x_{jk},$$

and hence each regression coefficient can now be written as $\beta_k = \sum_{j=1}^{n} \lambda_k x_{jk} w_j$, for which we see the λ_k's scaling role on the x_{jk}'s. Thus, the corresponding I-prior for $\boldsymbol{\beta}$ is

$$\boldsymbol{\beta} \sim \mathrm{N}_p(\mathbf{0}, \lambda^2 \mathbf{X}^\top \boldsymbol{\Lambda} \boldsymbol{\Psi} \boldsymbol{\Lambda} \mathbf{X}),$$

with $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$. Note that $\mathcal{F}_\lambda$ can be seen as a special case of the ANOVA RKKS, in which only the main effects are considered, in which case the *centred canonical RKHSs* should be considered instead. This approach is disadvantageous when p is large, in which case there would be numerous scale parameters to estimate.

Remark 4.1. The I-prior for $\boldsymbol{\beta}$ in (4.1) bears resemblance to the g-prior (Zellner, 1986), and in fact, the g-prior can be interpreted as an I-prior if the inner product of $\mathcal{X}$ is the Mahalonobis inner product. See Miscellanea 4.7.1 for a discussion.

4.1.2 Multilevel linear modelling

sec:multilevelmodels

Let $\mathcal{X} \equiv \mathbb{R}^p$, and suppose that alongside the covariates, there is information on group levels $\mathcal{M} = \{1, \ldots, m\}$ for each unit i. That is, every observation for unit i is known to belong to a specific group j, and we write $\mathbf{x}_i^{(j)}$ to indicate this. Let n_j denote the sample size for cluster j, and the overall sample size be $n = \sum_{j=1}^{m} n_j$. When modelled linearly with the responses $y_i^{(j)}$, the model is known as a multilevel (linear) model, although it is known by many other names: random-effects models, random coefficient models, hierarchical models, and so on. As this model is seen as an extension of linear models, applications are plenty, especially in research designs for which the data varies at more than one level.

Consider a functional ANOVA decomposition of the regression function as follows:

$$f(\mathbf{x}_i^{(j)}, j) = \alpha + f_1(\mathbf{x}_i^{(j)}) + f_2(j) + f_{12}(\mathbf{x}_i^{(j)}, j). \tag{4.2}$$

eq:anovamultilevel

To mimic the multilevel model, assume $f_1 \in \mathcal{F}_1$ the Pearson RKHS, $f_2 \in \mathcal{F}_2$ the centred canonical RKHS, and $f_{12} \in \mathcal{F}_{12} = \mathcal{F}_1 \otimes \mathcal{F}_2$, the tensor product space of $\mathcal{F}_1$ and $\mathcal{F}_2$. As we know, α is the overall intercept, and the varying intercepts are given by the function

f_2. While f_1 is the (main) linear effect of the covariates, f_{12} provides the varying linear effect of the covariates by each group. The I-prior for $f - \alpha$ is assumed to lie in the function space $\mathcal{F} - \alpha$, which is an ANOVA RKKS with kernel

$$h_\lambda\big((\mathbf{x}_i^{(j)}, j), (\mathbf{x}_i^{(j')}, j')\big) = \lambda_1 h_1(\mathbf{x}_i^{(j)}, \mathbf{x}_{i'}^{(j')}) + \lambda_2 h_2(j, j') + \lambda_1\lambda_2 h_1(\mathbf{x}_i^{(j)}, \mathbf{x}_{i'}^{(j')}) h_2(j, j'),$$

with h_1 the centred canonical kernel and h_2 the Pearson kernel. The reason for not including an RKHS of constant functions in $\mathcal{F}$ is because the overall intercept is usually simpler to estimate as an external parameter (see Section 4.2.1).

We can show that the regression function (4.2) corresponds to the standard way of writing the multilevel model,

$$f(\mathbf{x}_i^{(j)}, j) = \beta_0 + \mathbf{x}_i^{(j)\top}\boldsymbol{\beta}_1 + \beta_{0j} + \mathbf{x}_i^{(j)\top}\boldsymbol{\beta}_{1j}.$$

and determine the prior distributions on $(\beta_{0j}, \boldsymbol{\beta}_{1j}^\top)^\top \in \mathbb{R}^{p+1}$. For the interested reader, the details are in Miscellanea 4.7.2. The standard multilevel random effects assumption is that $(\beta_{0j}, \boldsymbol{\beta}_{1j}^\top)^\top$ is normally distributed with mean zero and covariance matrix $\boldsymbol{\Phi}$. In total, there are $p+1$ regression coefficients and $(p+1)(p+2)/2$ covariance parameters in Φ to be estimated. In contrast, the I-prior model is parameterised by only two RKKS scale parameters—one for $\mathcal{F}_1$ and one for $\mathcal{F}_2$—and the error precision ψ. While the estimation procedure for $\boldsymbol{\Phi}$ in the standard multilevel model can result in non-positive covariance matrices, the I-prior model has the advantage that positive definiteness is taken care of automatically[1].

As a remark, the following regression functions are nested

- $f(\mathbf{x}_i^{(j)}, j) = f_0 + f_1(\mathbf{x}_i^{(j)}) + f_2(j)$ (random intercept model);
- $f(\mathbf{x}_i^{(j)}, j) = f_0 + f_1(\mathbf{x}_i^{(j)})$ (linear regression model);
- $f(\mathbf{x}_i^{(j)}, j) = f_0 + f_2(j)$ (ANOVA model);
- $f(\mathbf{x}_i^{(j)}, j) = f_0$ (intercept only model),

and thus one may compare likelihoods to ascertain the best fitting model. In addition, one may add flexibility to the model in two possible ways:

1. **More than two levels**. The model can be easily adjusted to reflect the fact that

[1] By virtue of the estimate of the regression function belonging to $\mathcal{F}_n$, an RKHS with a positive definite kernel equal to the Fisher information for f.

that the data is structured in a hierarchy containing three or more levels. For the three level case, let the indices $j \in \{1, \ldots, m_1\}$ and $k \in \{1, \ldots, m_2\}$ denote the two levels, and simply decompose the regression function accordingly:

$$f(\mathbf{x}_i^{(j,k)}, j, k) = f_0 + f_1(\mathbf{x}_i^{(j,k)}) + f_2(j) + f_3(k) + f_{12}(\mathbf{x}_i^{(j,k)}, j) + f_{13}(\mathbf{x}_i^{(j,k)}, k) + f_{23}(j, k) + f_{123}(\mathbf{x}_i^{(j,k)}, j, k).$$

2. **Covariates not varying with levels**. Suppose now we would like to add covariates with a fixed effect to the model, i.e., covariates $\mathbf{z}_i^{(j)}$ which are not assumed to affect the responses differently in each group. The regression function would be:

$$f(\mathbf{x}_i^{(j)}, j, \mathbf{z}_j) = f_0 + f_1(\mathbf{x}_i^{(j)}) + f_2(j) + f_3(\mathbf{z}_i^{(j)}) + f_{12}(\mathbf{x}_i^{(j)}, j).$$

   This can be seen as a limited functional ANOVA decomposition of f.

Remark 4.2. Indexing can be tricky, but we find the following helpful. Supposing $m = 2$, and $n_1 = n_2 = 3$, then a typical panel data set looks like this:

y	x	z	i	j	k
y_{11}	x_{11}	z_1	1	1	1
y_{21}	x_{21}	z_1	2	1	2
y_{31}	x_{31}	z_1	3	1	3
y_{12}	x_{12}	z_2	1	2	4
y_{22}	x_{22}	z_2	2	2	5
y_{32}	x_{32}	z_2	3	2	6

The y's are the responses, x's covariates, and z's group-level covariates. If $\iota : (i, j) \mapsto k$ is a function which maps the dual index set (i, j) to the single index set $k \in \{1, \ldots, n\}$, then the multilevel regression function can be expressed as the regression function in model (1.1).

4.1.3 Longitudinal modelling

Longitudinal or panel data observes covariate measurements $x_i \in \mathcal{X}$ and responses $y_i(t) \in \mathbb{R}$ for individuals $i = 1, \ldots, n$ across a time period $t \in \{1, \ldots, T\} =: \mathcal{T}$. Often, the time indexing set $\mathcal{T}$ may be unique to each individual i, so measurements for unit i happens across a time period $\{t_{i1}, \ldots, t_{iT_i}\} =: \mathcal{T}_i$—this is known as an unbalanced panel. It is also possible that covariate measurements vary across time too, so appropriately they are denoted $x_i(t)$. For example, $x_i(t)$ could be repeated measurements of the variable x_i

at time point $t \in \mathcal{T}_i$. The relationship between the response variables $y_i(t)$ at time $t \in \mathcal{T}_i$ is captured through the equation

$$y_i(t) = f\big(x_i, t\big) + \epsilon_i(t)$$

where the distribution of $\boldsymbol{\epsilon}_i = \big(\epsilon_i(t_{i1}), \ldots, \epsilon_i(t_{iT_i})\big)^\top$ is Gaussian with mean zero and covariance matrix $\boldsymbol{\Psi}_i$. Assuming $\boldsymbol{\Psi}_i = \psi_i \mathbf{I}_{T_i}$ or even $\boldsymbol{\Psi}_i = \psi \mathbf{I}_{T_i}$ are perfectly valid choices, even though this seemingly ignores any time dependence between the observations. In reality, the I-prior induces time dependence of the observations via the kernels in the prior covariance matrix for f. Additionally, the random vectors $\boldsymbol{\epsilon}_i$ and $\boldsymbol{\epsilon}_{i'}$ are assumed to be independent for any two distinct $i, i' \in \{1, \ldots, n\}$.

Using the functional ANOVA decomposition on the regression function, we obtain

$$f(x_i, t) = f_0 + f_1(x_i) + f_2(t) + f_{12}(x_i, t), \tag{4.3}$$

{eq:longitudinalanova}

where f_0 is an overall constant, $f_1 \in \mathcal{F}_1$, $f_2 \in \mathcal{F}_2$, and $f_{12} \in \mathcal{F}_1 \otimes \mathcal{F}_2$. Choices for $\mathcal{F}_1$ and $\mathcal{F}_2$ are plentiful. In fact, any of the RKHS/RKKS described in Chapter 3 can be used to either model a linear dependence (canonical RKHS), nominal dependence (Pearson RKHS), polynomial dependence (polynomial RKKS) or smooth dependence (fBm or SE RKHS) on the x_i's and t's on f.

Remark 4.3. Although (4.3) is a special case of the multilevel model decomposition (4.2) for which $x_i = x_i(t)$ (time-varying covariates), it is different to how longitudinal models are normally treated using a mixed effects model. As a multilevel model, longitudinal models treat the individuals as the groups or clusters (level two), and the time points as the various measurements within the clusters (level one).

4.1.4 Smoothing models

Single- and multi-variable smoothing models can be fitted under the I-prior methodology using the fBm RKHS. In standard kernel based smoothing methods, the squared exponential kernel is often used, and the corresponding RKHS contains analytic functions. There are several attractive properties of using the fBm RKHS, and for one-dimensional smoothing, these are discussed below.

Assume that, up to a constant, the regression function lies in the scaled, centred fBm RKHS $\mathcal{F}$ of functions over $\mathcal{X} \equiv \mathbb{R}$ with Hurst index 1/2. Thus, with a centring with

respect to the empirical distribution P_n of $\{x_1, \ldots, x_n\}$ and using the absolute norm on $\mathbb{R}$, $\mathcal{F}$ has kernel

$$h_\lambda(x, x') = \frac{\lambda}{2n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \big(|x - x_i| + |x' - x_j| - |x - x'| - |x_i - x_j| \big).$$

According to van der Vaart and van Zanten (2008, Section 10), $\mathcal{F}$ contains absolutely continuous functions possessing a square integrable weak derivative satisfying $f(0) = 0$. The norm is given by $\|f\|_{\mathcal{F}}^2 = \int \dot{f}^2 \mathrm{d}x$. The posterior mean of f based on an I-prior is then a (one-dimensional) smoother for the data. For f of the form $f = \sum_{i=1}^{n} h(\cdot, x_i) w_i$, i.e., $f \in \mathcal{F}_n$, the finite subspace of $\mathcal{F}$ as in Section 3.4, then Bergsma (2017) shows that f can be represented as

$$f(x) = \int_{-\infty}^{x} \beta(t) \, \mathrm{d}t \tag{4.4}$$

{eq:ipriorbrownianbridge}

where

$$\beta(t) = \sum_{i : x_i \leq t} w_i = \frac{f(x_{i_t+1}) - f(x_{i_t})}{x_{i_t+1} - x_{i_t}} \tag{4.5}$$

with $i_t = \max_{x_i \leq t} i$. Under the I-prior with an iid assumption on the errors, the w_i's are zero mean normal random variables with variance ψ, so that β as defined above is an ordinary Brownian bridge with respect to the empirical distribution P_n. The I-prior for f is piecewise linear with knots at $x_1, \ldots, x_n$, and the same holds true for the posterior mean. The implication is that the I-prior automatically adapts to irregularly spaced x_i: in any region where there are no observations, the resulting smoother is linear. This is explained by the reduced Fisher information about the derivative of the regression curve in regions with no observation.

In Bergsma (2017), it is stated that the covariance function for β is

$$\mathrm{Cov}\big(\beta(x), \beta(x')\big) = n\big(\min\{\mathrm{P}_n(X < x), \mathrm{P}_n(X_n < x')\} - \mathrm{P}_n(X < x) \, \mathrm{P}_n(X_n < x') \big)$$

From this, notice that $\mathrm{Var}\, \beta(x) = \mathrm{P}_n(X_n < x)\big(1 - \mathrm{P}_n(X_n < x)\big)$, which shows an automatic boundary correction: close to the boundary there is little Fisher information on the derivative of the regression function $\beta(x)$, so the prior variance is small. This will lead to more shrinkage of the posterior derivative of f towards the derivative of the prior mean f_0.

Another advantage of the I-prior methodology is the ability to fit single or multidimensional smoothing models with just two parameters to be estimated: the RKHS scale parameter λ and the error precision $\boldsymbol{\Psi}$. The Hurst parameter $\gamma \in (0, 1)$ of the fBm RKHS can also be treated as a free parameter for added flexibility, but for most practical applications, we find that the default setting of $\gamma = 1/2$ performs sufficiently well.

Remark 4.4. From (4.4), the prior process for f is thus an integrated Brownian bridge. This shows a close relation with cubic spline smoothers, which can be interpreted as the posterior mean when the prior is an integrated Wiener process (Wahba, 1990). Unlike I-priors however, cubic spline smoothers do not have automatic boundary corrections, and typically the additional assumption is made that the smoothing curve is linear at the boundary knots.

4.1.5 Regression with functional covariates

sec:regfunctionalcov

Suppose that we have functional covariates x in the real domain, and that $\mathcal{X}$ is a set of differentiable functions. If so, it is reasonable to assume that $\mathcal{X}$ is a Hilbert-Sobolev space with inner product

$$\langle x, x' \rangle_{\mathcal{X}} = \int \dot{x}(t)\dot{x}'(t)\,\mathrm{d}t,$$

so that we may apply the linear, fBm or any other kernels which make use of inner products by making use of the polarisation identity. Furthermore, let $z \in \mathbb{R}^T$ be the discretised realisation of the function $x \in \mathcal{X}$ at regular intervals $t = 1, \ldots, T$. Then

$$\langle x, x' \rangle_{\mathcal{X}} \approx \sum_{t=1}^{T-1} (z_{t+1} - z_t)(z'_{t+1} - z'_t).$$

For discretised observations at non-regular intervals $\{t_1, \ldots, t_T\}$ then a more general formula to the above one might be used, for instance,

$$\langle x, x' \rangle_{\mathcal{X}} \approx \sum_{i=1}^{T-1} \frac{(z_{t_{i+1}} - z_{t_i})(z'_{t_{i+1}} - z'_{t_i})}{t_{i+1} - t_i}.$$

4.2 Estimation

sec:ipriorestimation

After selecting a RKHS/RKKS $\mathcal{F}$ of functions over $\mathcal{X}$ suitable for the regression problem at hand, one then proceeds to estimate the posterior distribution of the regression func-

tion. The I-prior model (1.1) subject to (1.2) and $f \in \mathcal{F}$ has the simple and convenient representation

$$\begin{aligned} y_i &= \alpha + f_0(x_i) + \overbrace{\sum_{k=1}^{n} h_\eta(x_i, x_k) w_k}^{f(x_i)} + \epsilon_i \\ (\epsilon_1, \dots, \epsilon_n)^\top &\sim \mathrm{N}_n(\mathbf{0}, \boldsymbol{\Psi}^{-1}) \\ (w_1, \dots, w_n)^\top &\sim \mathrm{N}_n(\mathbf{0}, \boldsymbol{\Psi}), \end{aligned} \tag{4.6}$$

{eq:model2}

where $f_0 : \mathcal{X} \to \mathbb{R}$ is a function chosen a priori representing the 'best guess' of f, and the dependence of the kernel of $\mathcal{F}$ on parameters η is emphasised through the subscript in $h_\eta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

The parameters of the I-prior model are collectively denoted by $\theta = \{\alpha, \eta, \boldsymbol{\Psi}\}$. Given θ and a prior choice for f_0, the posterior regression function is determined solely by the posterior distribution of the w_i's. Using standard multivariate normal results, one finds that the posterior distribution for $\mathbf{w} := (w_1, \dots, w_n)^\top$ is $\mathbf{w}|\mathbf{y} \sim \mathrm{N}_n(\tilde{\mathbf{w}}, \tilde{\mathbf{V}}_w)$, where

$$\tilde{\mathbf{w}} = \boldsymbol{\Psi}\mathbf{H}_\eta \mathbf{V}_y^{-1}(\mathbf{y} - \alpha\mathbf{1}_n - \mathbf{f}_0) \quad \text{and} \quad \tilde{\mathbf{V}}_w = \left(\mathbf{H}_\eta \boldsymbol{\Psi} \mathbf{H}_\eta + \boldsymbol{\Psi}^{-1}\right)^{-1} = \mathbf{V}_y^{-1}, \tag{4.7}$$

{eq:posteriorw}

using the familiar notation that we introduced in Section 1.4. For a derivation, see Section 4.8. By linearity, the posterior distribution for f is also normal.

In each modelling scenario, there are a number of kernel parameters η that need to be estimated from the data. Assuming that the covariate space is $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_p$, and there is an ANOVA like decomposition of the function space $\mathcal{F}$ into its constituents spaces $\mathcal{F}_1, \dots, \mathcal{F}_p$, then at the very least, there are p scale parameters $\lambda_1, \dots, \lambda_p$ for each of the RKHSs. Depending on the RKHS used, there could be more kernel parameters that need to be optimised, for instance, the Hurst index for the fBm RKHS, the lengthscale for the SE RKHS, and/or the offset for the polynomial RKKS. However, these may be treated as fixed parameters as well.

The following subsections describe possible estimation procedures for the hyperparameters of the model. Henceforth, for simplicity, the following additional standing assumptions are imposed on the I-prior model (4.6):

A1 **Centred responses**. Set $\alpha = 0$ and replace the responses by their centred versions $y_i \mapsto \tilde{y}_i = y_i - \frac{1}{n}\sum_{i=1}^{n}$.

ass:A1

ass:A2

A2 **Zero prior mean**. Assume a zero prior mean $f_0(x) = 0$ for all $x \in \mathcal{X}$.

ass:A3

A3 **Iid errors**. Assume identical and independent (iid) errors random variables, i.e., $\boldsymbol{\Psi} = \psi \mathbf{I}_n$.

Assumptions A1 and A2 are motivated by the discussion in Section 4.2.1. Although assumption A3 is not strictly necessary, it is often a reasonable one and one that simplifies the estimation procedure greatly.

sec:interce
pt

4.2.1 The intercept and the prior mean

In most statistical models, an intercept is a necessary inclusion which aids interpretation. In the context of the I-prior model (4.6), a lack of an intercept would fail to account for the correct locational shift of the regression function along the y-axis. Further, when zero-mean functions are considered, the intercept serves as being the 'grand mean' value of the responses.

The addition of an intercept to the regression model may be viewed in one of two ways. The first is to view it as a function belonging to the RKHS of constant functions $\mathcal{F}_0$, and thereby tensor summing this space to $\mathcal{F}$. In the polynomial and ANOVA RKKSs, we saw that an intercept is naturally induced by the inclusion of a RKHS of constant functions in their construction. The second is to simply treat the intercept as a parameter of the model to be estimated. In any of the other RKHSs described in Chapter 2, an intercept would need to be added separately.

These two methods convey the same mathematical model, and there is very little difference in the way of interpretation, although estimation is entirely different. In the first method, the intercept-less RKHS/RKKS $\mathcal{F}$ with kernel h is made to include an intercept by modifying the kernel to be $h+1$. The intercept will then be implicitly taken care of without having dealt with it explicitly. However, it can be obtained by realising that for $\alpha \in \mathcal{F}_0$ the RKHS of constant functions, then $\alpha = \sum_{i=1}^n w_i$.

On the other hand, consider the intercept as a parameter α to be estimated. Obtaining an estimate α using a likelihood-based argument is rather simple. From (4.6), $\mathrm{E}\, y_i = \alpha + f_0(x_i)$ for all $i = 1, \ldots, n$, so the maximum likelihood estimate for $\mathrm{E}\, y$ is its sample mean $\bar{y} = \frac{1}{n}\sum_{i=1} y_i$, and hence the ML estimate for α is $\hat{\alpha} = \bar{y} - \frac{1}{n}\sum_{i=1}^n f_0(x_i)$. Alternatively, the estimation of α under a fully Bayesian treatment is possible by assuming an appropriate hyperprior on it, such as a conjugate normal prior $\mathrm{N}(a, A^{-1})$. If

so, the conditional posterior of α given $\mathbf{w}$, η, $\boldsymbol{\Psi}$ and f_0 is also normal with mean $\tilde{a}$ and variance $\tilde{A}$, where

$$\tilde{A} = \sum_{i,j=1}^{n} \psi_{ij} + A \quad \text{and} \quad \tilde{a} = \tilde{A}^{-1} \left(\sum_{i=1}^{n} [(\mathbf{y} - \mathbf{f}_0 - \mathbf{H}_\eta \mathbf{w}) \boldsymbol{\Psi}]_i + Aa \right).$$

This fact can be used, say, in conjunction with a Gibbs sampling procedure treating the rest of the unknowns as random. Note that the posterior mean for α is

$$\mathrm{E}[\alpha|\mathbf{y}] = \mathrm{E}_{\mathbf{w}} \left[\mathrm{E}[\alpha|\mathbf{y}, \mathbf{w}] \right] = \frac{\sum_{i,j=1}^{n} \psi_{ij}(y_i - f_0(x_i)) + Aa}{\sum_{i,j=1}^{n} \psi_{ij} + A},$$

which, in the iid errors case, is seen to be a weighted sum of the ML estimate $\hat{\alpha}$ and the prior mean a. Unless there is a strong reason to add prior information to the intercept, the ML estimate seems to be the simplest approach. Assumption A1 implies a ML estimation of the intercept parameter.

Now, a note on the prior mean f_0. For kernels with the property that $h(x, x^*) \to 0$ as $D(x, x^*) \to \infty$ for $x \in \mathcal{X}_{\text{train}}$ and $x^* \in \mathcal{X}_{\text{new}}$ such as the SE kernel, this means that predictions outside the training set will be zero and thus rely on the prior mean f_0. However, all of the other kernels in this thesis, namely the fBm, canonical, and polynomial kernels, do not have this property—they instead use information provided by the training data to extrapolate predictions far away from the data set. A prior mean of zero seems reasonable and safe in the absence of any prior information, so long as the global and local properties of the regression function are understood with respect to the kernel chosen. $f_0 = 0$ also implies a complete reliance on the data rather than subjective prior belief of a suitable choice for f.

Of course, should it be felt appropriate, a non-zero function f_0 may be imposed as the prior mean. If $f_0(x) = \mu_0 \in \mathbb{R}$ for all $x \in \mathcal{X}$, then this basically implies another intercept in the model, if it is not already present. Note that when treating μ_0 as a hyperparameter to be estimated, then this does not yield a fully identified model, and only $\alpha + \mu_0$ may be estimated.

4.2.2 Direct optimisation

Under assumptions A1 and A2, a direct optimisation of the parameters $\theta = \{\eta, \boldsymbol{\Psi}\}$ using the log-likelihood of θ is straightforward to implement. Denote $\boldsymbol{\Sigma}_\theta := \mathbf{H}_\eta \boldsymbol{\Psi} \mathbf{H}_\eta + \boldsymbol{\Psi}^{-1} =$

$\mathbf{V}_y$. From (4.6), the (marginal) log-likelihood of θ is given by

$$\begin{aligned} L(\theta) &= \log \int p(\mathbf{y}|\mathbf{w}) p(\mathbf{w}) \, \mathrm{d}\mathbf{w} \\ &= -\frac{n}{2} \log 2\pi - \frac{1}{2} \log |\boldsymbol{\Sigma}_\theta| - \frac{1}{2} \tilde{\mathbf{y}}^\top \boldsymbol{\Sigma}_\theta^{-1} \tilde{\mathbf{y}}. \end{aligned} \tag{4.8}$$

{eq:marglog liky}

The term marginal refers to the fact that we are averaging out the random function represented by $\mathbf{w}$. Direct optimisation is typically done using conjugate gradients with a Cholesky decomposition on the covariance kernel to maintain stability, but we opt for an eigendecomposition of the kernel matrix $\mathbf{H}_\eta = \mathbf{V} \cdot \mathrm{diag}(u_1, \ldots, u_n) \cdot \mathbf{V}^\top$ instead. Further, under assumption A3 and since $\mathbf{H}_\eta$ is a symmetric matrix, we have that $\mathbf{V}\mathbf{V}^\top = \mathbf{I}_n$, and thus

$$\mathbf{V}_y = \mathbf{V} \cdot \mathrm{diag}(\psi u_1^2 + \psi^{-1}, \ldots, \psi u_n^2 + \psi^{-1}) \cdot \mathbf{V}^\top$$

for which the inverse and log-determinant is easily obtainable. This method is relatively robust to numerical instabilities and is better at ensuring positive definiteness of the covariance kernel. The eigendecomposition is performed using the **Eigen** C++ template library and linked to **iprior** using **Rcpp** (Eddelbuettel and Francois, 2011). The hyperparameters are transformed by the **iprior** package so that an unrestricted optimisation using the quasi-Newton L-BFGS algorithm provided by `optim()` in R. Note that minimisation is done on the deviance scale, i.e., minus twice the log-likelihood. The direct optimisation method can be prone to local optima, in which case repeating the optimisation at different starting points and choosing the one which yields the highest likelihood is one way around this.

2. Show ridge in the log-likelihood plot.

Let $\mathbf{U}$ be the Fisher information matrix for $\theta \in \mathbb{R}^q$. Standard calculations (Section 4.11) show that under the marginal distribution $\tilde{\mathbf{y}} \sim \mathrm{N}_n\left(\mathbf{0}, \boldsymbol{\Sigma}_\theta\right)$, the (i,j)th coordinate of $\mathbf{U}$ is

$$u_{ij} = \frac{1}{2} \mathrm{tr}\left(\boldsymbol{\Sigma}_\theta^{-1} \frac{\partial \boldsymbol{\Sigma}_\theta}{\partial \theta_i} \boldsymbol{\Sigma}_\theta^{-1} \frac{\partial \boldsymbol{\Sigma}_\theta}{\partial \theta_j} \right)$$

where the derivative of a matrix with respect to a scalar is the element-wise derivative of the matrix. With $\hat{\theta}$ denoting the ML estimate for θ, under suitable conditions, $\sqrt{n}(\hat{\theta} - \theta)$ has an asymptotic multivariate normal distribution with mean zero and covariance matrix $\mathbf{U}^{-1}$ (Casella and R. L. Berger, 2002). In particular, the standard errors for θ_k are the diagonal elements of $\mathbf{U}^{-1/2}$.

4.2.3 Expectation-maximisation algorithm

Evidently, (4.6) lends itself to resembling a random-effects model, for which the EM algorithm can easily be employed to estimate its hyperparameters. Assume A1 and A2 holds. By treating the complete data as $\{\mathbf{y}, \mathbf{w}\}$ and the w_i's as "missing", the tth iteration of the E-step entails computing

$$\begin{aligned} Q(\theta) &= \mathrm{E}_{\mathbf{w}}\left[\log p(\mathbf{y}, \mathbf{w}|\theta) \middle| \mathbf{y}, \theta^{(t)}\right] \\ &= \mathrm{E}_{\mathbf{w}}\left[\text{const.} - \frac{1}{2}(\tilde{\mathbf{y}} - \mathbf{H}_\eta \mathbf{w})^\top \boldsymbol{\Psi}(\tilde{\mathbf{y}} - \mathbf{H}_\eta \mathbf{w}) - \frac{1}{2}\mathbf{w}^\top \boldsymbol{\Psi}^{-1}\mathbf{w} \,\middle|\, \mathbf{y}, \theta^{(t)}\right] \\ &= \text{const.} - \frac{1}{2}\tilde{\mathbf{y}}^\top \boldsymbol{\Psi}\tilde{\mathbf{y}} - \frac{1}{2}\operatorname{tr}\left(\overbrace{(\mathbf{H}_\eta \boldsymbol{\Psi}\mathbf{H}_\eta + \boldsymbol{\Psi}^{-1})}^{\boldsymbol{\Sigma}_\theta}\tilde{\mathbf{W}}^{(t)}\right) + \tilde{\mathbf{y}}^\top \boldsymbol{\Psi}\mathbf{H}_\eta \tilde{\mathbf{w}}^{(t)}, \end{aligned} \tag{4.9}$$

{eq:QfnEstep}

where $\tilde{\mathbf{w}}^{(t)} = \mathrm{E}[\mathbf{w}|\mathbf{y}, \theta^{(t)}]$ and $\tilde{\mathbf{W}}^{(t)} = \mathrm{E}[\mathbf{w}\mathbf{w}^\top|\mathbf{y}, \theta^{(t)}]$ are the first and second posterior moments of $\mathbf{w}$ calculated at the tth EM iteration. These can be computed directly from (4.7), substituting for $\theta^{(t)} = \{\eta^{(t)}, \boldsymbol{\Psi}^{(t)}\}$ as appropriate. Note that (4.9) follows as a direct consequence of the results in Section 4.8.

Now, assume that A3 holds. The M-step then assigns $\theta^{(t+1)}$ the value of θ which maximises the Q function above. This boils down to solving the first order conditions

$$\frac{\partial Q}{\partial \eta} = -\frac{1}{2}\operatorname{tr}\left(\frac{\partial \boldsymbol{\Sigma}_\theta}{\partial \eta}\tilde{\mathbf{W}}^{(t)}\right) + \psi \cdot \tilde{\mathbf{y}}^\top \frac{\partial \mathbf{H}_\eta}{\partial \eta}\tilde{\mathbf{w}}^{(t)} \tag{4.10}$$

{eq:emtheta}

$$\frac{\partial Q}{\partial \psi} = -\frac{1}{2}\tilde{\mathbf{y}}^\top \tilde{\mathbf{y}} - \operatorname{tr}\left(\frac{\partial \boldsymbol{\Sigma}_\theta}{\partial \psi}\tilde{\mathbf{W}}^{(t)}\right) + \tilde{\mathbf{y}}^\top \mathbf{H}_\eta \tilde{\mathbf{w}}^{(t)} \tag{4.11}$$

{eq:empsi}

equated to zero. As $\partial \boldsymbol{\Sigma}_\theta / \partial \psi = \mathbf{H}_\eta^2 - \psi^{-2}\mathbf{I}_n$, the solution to (4.11) for ψ is separable in η. Meaning, given values for η, the solution $\psi^{(t+1)}$ emits a closed form

$$\psi^{(t+1)} = \left\{\frac{\operatorname{tr}\tilde{\mathbf{W}}^{(t)}}{\tilde{\mathbf{y}}^\top\tilde{\mathbf{y}} + \operatorname{tr}(\mathbf{H}_\eta^2\tilde{\mathbf{W}}^{(t)}) - 2\tilde{\mathbf{y}}^\top\mathbf{H}_\eta\tilde{\mathbf{w}}^{(t)}}\right\}^{1/2}. \tag{4.12}$$

{eq:closedformpsi}

We use this fact to form a sequential updating scheme $\eta^{(t)} \to \psi^{(t+1)} \to \eta^{(t+1)} \to \cdots$, and this form of the EM algorithm is known as the *expectation conditional maximisation* algorithm (Meng and Rubin, 1993). The solution to (4.10) can also be found in closed-form given values ψ, for many models, but in general, this is not the case. In cases where closed-form solutions do exist for η, then it is just a matter of iterating the update equations until a suitable convergence criterion is met (e.g. no more sizeable increase in

successive log-likelihood values). In cases where closed-form solutions do not exist for η, the Q function is again optimised with respect to η using the L-BFGS algorithm.

In our experience, the EM algorithm is more stable than direct maximisation, in the sense that the EM steps increase the likelihood in a gentle manner that prevents sudden explosions of the likelihood. The reason for this is that the Q function is generally convex in the parameters (at the very least, it is convex in each coordinate of θ, in most cases anyway). As such, the EM is especially suitable if there are many scale parameters to estimate, but on the flip side, it is typically slow to converge. The **iprior** package provides a method to automatically switch to the direct optimisation method after running several EM iterations. This then combines the stability of the EM with the speed of direct optimisation.

4.2.4 Markov chain Monte Carlo methods

For completeness, it should be mentioned that a full Bayesian treatment of the model is possible, with additional priors on the set of hyperparameters. Markov chain Monte Carlo (MCMC) methods can then be employed to sample from the posteriors of the hyperparameters, with point estimates obtained using the posterior mean or mode, for instance. Additionally, the posterior distribution encapsulates the uncertainty about the parameter, for which inference can be made. Posterior sampling can be done using Gibbs-based methods in **WinBUGS** (Lunn et al., 2000) or **JAGS** (Plummer, 2003), and both have interfaces to R via **R2WinBUGS** (Sturtz et al., 2005) and **runjags** (Denwood, 2016) respectively. Hamiltonian Monte Carlo (HMC) sampling is also a possibility, and the Stan project (Carpenter et al., 2017) together with the package **rstan** (Stan Development Team, 2016) makes this possible in R.

On the software side, all of these MCMC packages require the user to code the model individually, and we are not aware of the existence of MCMC-based packages which are able to estimate GPR models. This makes it inconvenient for GPR and I-prior models, because in addition to the model itself, the kernel functions need to be coded as well and ensuring computational efficiency would be a difficult task.

Speaking of efficiency, it is more advantageous to marginalise the I-prior and work with the marginal model (4.8), rather than the hierarchical specification (4.6). The reason for this is that the latter model has a parameter space whose dimension is $O(n)$, while the former only samples the hyperparameters. The posterior sampling for the w_i's

in (equivalently, the posterior Gaussian process $f(x) = \sum_{i=1}^{n} h_\lambda(x, x_i) w_i$) is performed using the normal posterior distribution in (4.7).

4.2.5 Comparison of estimation methods

sec:compare estimation

Consider a one-dimensional smoothing example. $n = 150$ data pairs (y_i, x_i) have been randomly sampled according to the true relationship

$$r_i = \overbrace{\text{const.} + 0.35 \cdot \phi(x_i|1, 0.8^2) + 0.65 \cdot \phi(x_i|4, 1.5^2) + \mathbb{1}(x_i > 4.5) \cdot e^{1.25(x_i - 4.5)}}^{f_{\text{true}}(x_i)}, \tag{4.13}$$

eq:example smoothingdata

where $\phi(\cdot|\mu, \sigma^2)$ is the probability density function of the normal distribution with mean μ and variance σ^2. The observed y_i's are thought to be noisy versions of the true points, i.e. $y_i = r_i + \epsilon_i$, with ϵ_i following an indescript, not necessarily normal, distribution. The predictors $x_1, \ldots, x_n$ have been sampled roughly from the interval $(-1, 6)$, and the sampling was intentionally not uniform so that there is slight sparsity in the middle. Figure 4.1 plots the sampled points and the true regression function.

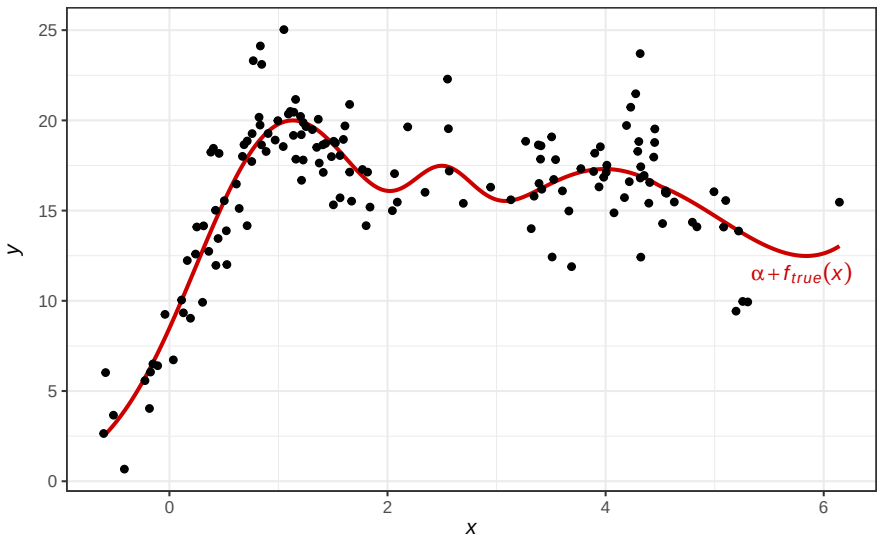


Figure 4.1: A plot of the sampled data points according to equation (4.13), with the true regression function superimposed.

fig:example smoothingdata

We attempt to estimate f_{true} by a function f belonging to the fBm-0.5 RKHS $\mathcal{F}_\lambda$, with an I-prior on f. There are two parameters that need to be estimated: the scale

parameter λ for the fBm-0.5 RKHS, and the error precision ψ. These can be estimated using the maximum likelihood methods described above, namely by direct optimisation and the EM algorithm. These two methods are implemented in the **iprior** package. A full Bayesian treatment is possible, and we use the **rstan** implementation of Stan to perform Hamiltonian Monte Carlo sampling of the posterior densities. A vague prior choice for λ and ψ are prescribed, namely

$$\lambda, \psi \overset{\text{iid}}{\sim} \text{N}_+(0, 100),$$

where $\text{N}_+(\mu, \sigma^2)$ represents the *half-normal* distribution[2]. We have also set an improper prior density $p(\alpha) \propto$ const. for the intercept. The advantage of HMC is that efficiency is not dictated by conjugacy, so there is freedom to choose any appropriate prior choice on the parameters.

Table 4.1: Table comparing the estimated parameter values, (marginal) log-likelihood values, and also time taken for the three estimation methods.

tab:compare methodsesti mate

	Direct optimisation	EM algorithm	Hamiltonian MC
Intercept (α)	16.1 (NA)	16.1 (NA)	16.1 (0.17)
Scale (λ)	5.01 (1.23)	5.01 (1.26)	5.61 (1.42)
Precision (ψ)	0.236 (0.03)	0.236 (0.03)	0.237 (0.03)
Log density	-339.7	-339.7	-341.1
Predictive RMSE	0.574	0.575	0.582
Iterations	12	266	2000
Time taken (s)	0.96	3.65	232

Table 4.1 tabulates the estimated parameter values, (marginal) log-likelihood values, and also time taken for the three estimation methods. The three methods concur on the estimated parameter values, although the scale parameter has been estimated slightly differently, which is possibly attributed to the effect of the prior for λ. The resulting log-likelihood value for the Bayesian method is lower than the ML methods, which also took the longest to compute. Although the EM algorithm took longer than the direct optimisation method to compute, the time taken per iteration is significantly shorter than one Newton iteration.

[2]The random variable $X \sim \text{N}_+(\mu, \sigma^2)$ has the density $p(x) = \phi(x|\mu, \sigma^2)\,\mathbb{1}(x \geq 0)$.

sec:ipriorcompcons

4.3 Computational considerations

Computational complexity for estimating I-prior models (and in fact, for GPR in general) is dominated by the inversion (by way of eigendecomposition in our case) of the $n \times n$ matrix $\boldsymbol{\Sigma}_\theta = \mathbf{H}_\eta \boldsymbol{\Psi} \mathbf{H}_\eta + \boldsymbol{\Psi}^{-1}$, which scales as $O(n^3)$ in time. For the direct optimisation method, this matrix inversion is called when computing the log-likelihood, and thus must be computed at each Newton step. For the EM algorithm, this matrix inversion appears when calculating $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{W}}$, the first and second posterior moments of the I-prior random effects. Furthermore, storage requirements for I-priors models are similar to that of GPR models, which is $O(n^2)$. In what follows, assumptions A1–A3 hold.

4.3.1 The Nyström approximation

The shared computational issues of I-prior and GPR models allow us to delve into machine learning literature, which is rich in ways to resolve these issue, as summarised by Quiñonero-Candela and Rasmussen (2005). One such method is to exploit low rank structures of kernel matrices. The idea is as follows. Let $\mathbf{Q}$ be a matrix with rank $q < n$, and suppose that $\mathbf{Q}\mathbf{Q}^\top$ can be used sufficiently well to represent the kernel matrix $\mathbf{H}_\eta$. Then

$$(\psi \mathbf{H}_\eta^2 + \psi^{-1} \mathbf{I}_n)^{-1} \approx \psi \left[\mathbf{I}_n - \mathbf{Q} \left(\left(\psi^2 \mathbf{Q}^\top \mathbf{Q} \right)^{-1} + \mathbf{Q}^\top \mathbf{Q} \right)^{-1} \mathbf{Q}^\top \right],$$

obtained via the Woodbury matrix identity, is potentially a much cheaper operation which scales $O(nq^2)$: $O(q^3)$ to do the inversion, and $O(nq)$ to do the multiplication (because typically the inverse is premultiplied to a vector). When using the linear kernel for a low-dimensional covariate then the above method is exact. This fact is clearly demonstrated by the equivalence of the p-dimensional linear model implied by (4.1) with the n-dimensional I-prior model using the canonical RKHS. If $p \ll n$ then certainly using the linear representation is much more efficient.

However, other interesting kernels such as the fractional Brownian motion (fBm) kernel or the squared exponential kernel results in kernel matrices which are full rank. An approximation to the kernel matrix using a low-rank matrix is the Nyström method (Williams and Seeger, 2001). The theory has its roots in approximating eigenfunctions, but this has since been adopted to speed up kernel machines. The main idea is to obtain an (approximation to the true) eigendecomposition of $\mathbf{H}_\eta$ based on a small subset $m \ll n$ of the data points.

Let $\mathbf{H}_\eta = \mathbf{V}\mathbf{U}\mathbf{V}^\top = \sum_{i=1}^n u_i \mathbf{v}_i \mathbf{v}_i^\top$ be the (orthogonal) decomposition of the symmetric matrix $\mathbf{H}_\eta$. As mentioned, avoiding this expensive $O(n^3)$ eigendecomposition is desired, and this is achieved by selecting a subset $\mathcal{M}$ of size m of the n data points $\{1, \ldots, n\}$, so that $\mathbf{H}_\eta$ may be approximated using the rank m matrix $\mathbf{H}_\eta \approx \sum_{i \in \mathcal{M}} \tilde{u}_i \tilde{\mathbf{v}}_i \tilde{\mathbf{v}}_i^\top$. Without loss of generality, reorder the rows and columns of $\mathbf{H}_\eta$ so that the data points indexed by $\mathcal{M}$ are used first:

$$\mathbf{H}_\eta = \begin{pmatrix} \mathbf{A}_{m \times m} & \mathbf{B}_{m \times (n-m)} \\ \mathbf{B}^\top_{m \times (n-m)} & \mathbf{C}_{(n-m) \times (n-m)} \end{pmatrix}.$$

In other words, the data points indexed by $\mathcal{M}$ forms the smaller $m \times m$ kernel matrix $\mathbf{A}$. Let $\mathbf{A} = \mathbf{V}_m \mathbf{U}_m \mathbf{V}_m^\top = \sum_{i=1}^m u_i^{(m)} \mathbf{v}_i^{(m)} \mathbf{v}_i^{(m)\top}$ be the eigendeceomposition of $\mathbf{A}$. The Nyström method provides the formulae for $\tilde{u}_i$ and $\tilde{\mathbf{v}}_i$ (Rasmussen and Williams, 2006, §8.1, equations 8.2 and 8.3) as

$$\begin{aligned} \tilde{u}_i &:= \frac{n}{m} u_i^{(m)} \in \mathbb{R} \\ \tilde{\mathbf{v}}_i &:= \sqrt{\frac{m}{n}} \frac{1}{u_i^{(m)}} \begin{pmatrix} \mathbf{A} & \mathbf{B} \end{pmatrix}^\top \mathbf{v}_i^{(m)} \in \mathbb{R}^n. \end{aligned}$$

Denoting $\mathbf{U}_m$ as the diagonal matrix of eigenvalues $u_1^{(m)}, \ldots, u_m^{(m)}$, and $\mathbf{V}_m$ the corresponding matrix of eigenvectors $\mathbf{v}_i^{(m)}$, we have

$$\mathbf{H}_\eta \approx \overbrace{\begin{pmatrix} \mathbf{V}_m \\ \mathbf{B}^\top \mathbf{V}_m \mathbf{U}_m^{-1} \end{pmatrix}}^{\bar{\mathbf{V}}} \mathbf{U}_m \overbrace{\begin{pmatrix} \mathbf{V}_m^\top & \mathbf{U}_m^{-1} \mathbf{V}_m^\top \mathbf{B} \end{pmatrix}}^{\bar{\mathbf{V}}^\top}.$$

Unfortunately, it may be the case that $\bar{\mathbf{V}}\bar{\mathbf{V}}^\top \neq \mathbf{I}_n$, while orthogonality is crucial in order to easily calculate the inverse of $\boldsymbol{\Sigma}_\theta$. An additional step is required to obtain an orthogonal version of the Nyström decomposition, as studied by Fowlkes et al. (2001). Let $\mathbf{K} = \mathbf{A} + \mathbf{A}^{-\frac{1}{2}} \mathbf{B}^\top \mathbf{B} \mathbf{A}^{-\frac{1}{2}}$, where $\mathbf{A}^{-\frac{1}{2}} = \mathbf{V}_m \mathbf{U}_m^{-\frac{1}{2}} \mathbf{V}_m$, and obtain the eigendecomposition of this $m \times m$ matrix $\mathbf{K} = \mathbf{R}\hat{\mathbf{U}}\mathbf{R}^\top$. Defining

$$\hat{\mathbf{V}} = \begin{pmatrix} \mathbf{A} \\ \mathbf{B}^\top \end{pmatrix} \mathbf{A}^{-\frac{1}{2}} \mathbf{R} \hat{\mathbf{U}}^{-\frac{1}{2}} \in \mathbb{R}^n \times \mathbb{R}^m,$$

then we have that $\mathbf{H}_\eta \approx \hat{\mathbf{V}}\hat{\mathbf{U}}\hat{\mathbf{V}}^\top$ such that $\hat{\mathbf{V}}\hat{\mathbf{V}}^\top = \mathbf{I}_n$. Estimating I-prior models with the Nyström method including the orthogonalisation step takes roughly $O(nm^2)$ time

3. Attempt to prove this.

and $O(nm)$ storage.

The issue of selecting the subset $\mathcal{M}$ remains. The simplest method, and that which is implemented in the **iprior** package, would be to uniformly sample a subset of size m from the n points. Although this works well in practice, the quality of approximation might suffer if the points do not sufficiently represent the training set. In this light, greedy approximations have been suggested to select the m points, so as to reduce some error criterion relating to the quality of approximation. For a brief review of more sophisticated methods of selecting $\mathcal{M}$, see Rasmussen and Williams (2006, §8.1, pp. 173–174).

4.3.2 An efficient EM algorithm

sec:efficientEM1

The evaluation of the Q function in (4.9) is $O(n^3)$, because a change in the values of θ requires evaluating $\mathbf{\Sigma}_\theta = \psi\mathbf{H}_\eta^2 + \psi^{-1}\mathbf{I}_n$, for which squaring $\mathbf{H}_\eta$ takes the bulk of the computational time. In this section, we describe an efficient method of evaluating Q if the I-prior model only involves estimating the RKHS scale parameters and the error precision under assumptions A1–A3.

Corresponding to p building block RKHSs $\mathcal{F}_1, \dots, \mathcal{F}_p$ of functions over $\mathcal{X}_1, \dots, \mathcal{X}_p$, there are p scale parameters $\lambda_1, \dots, \lambda_p$ and reproducing kernels $h_1, \dots, h_p$. Write $\theta = \{\lambda_1, \dots, \lambda_p, \psi\}$. The most common modelling scenarios that will be encountered are listed below:

1. **Single scale parameter**. With $p = 1$, $f \in \mathcal{F} \equiv \lambda_1\mathcal{F}_1$ of functions over a set $\mathcal{X}$. $\mathcal{F}$ may be any of the building block RKHSs. Note that $\mathcal{X}_1$ itself may be more than one-dimensional. The kernel over $\mathcal{X}_1 \times \mathcal{X}_1$ is therefore
$$h_\lambda = \lambda_1 h_1.$$

2. **Multiple scale parameters**. Here, $\mathcal{F}$ is a RKKS of functions $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_p \to \mathbb{R}$, and thus $\mathcal{F} \equiv \lambda_1\mathcal{F}_1 \oplus \cdots \oplus \lambda_p\mathcal{F}_p$, where each $\mathcal{F}_k$ is one of the building block RKHSs. The kernel is
$$h_\lambda = \lambda_1 h_1 + \cdots + \lambda_p h_p.$$

3. **Multiple scale parameters with level-2 interactions**. This occurs commonly with multilevel and longitudinal models. Suppose that $\mathcal{X}_1$ is the set of 'levels' and there are $p - 1$ covariate sets $\mathcal{X}_k$, $k = 2, \cdots, p$. The function space $\mathcal{F}$ is a special

case of the ANOVA RKKS containing only main and two-way interaction effects, and its kernel is

$$h_\lambda = \sum_{j=1}^{p} \lambda_j h_j + \sum_{j<k} \lambda_j \lambda_k h_j h_k,$$

where $\mathcal{F}_1$ is the Pearson RKHS, and the remaining are any of the building block RKHSs.

4. **Polynomial RKKS**. When using the polynomial RKKS of degree d to incite a polynomial relationship of the covariate set $\mathcal{X}_1$ on the function $f \in \mathcal{F}$ (excluding an intercept), then the kernel of $\mathcal{F}$ is

$$h_\lambda = \sum_{k=1}^{d} b_k \lambda_1^k h_1^k.$$

where $b_k = \frac{d!}{k!(d-k)!}$, $k = 1, \ldots, d$ are constants.

Of course, many other models are possible, such as the ANOVA RKKS with all p levels of interactions. What we realise is that any of these scenarios are simply a sum-product of a manipulation of the set of scale parameters $\lambda = \{\lambda_1, \ldots, \lambda_p\}$ and the set of kernel functions $h = \{h_1, \ldots, h_p\}$.

Let us be more concrete about what we mean by 'manipulation' of the sets λ and h. Define an 'instruction operator' which expands out both sets identically as required by the modelling scenario. Computationally speaking, this instruction could be as simple as a list containing the indices to multiply out. For the four scenarios above, the list $\mathcal{Q}$ is

1. $\mathcal{Q} = \big\{\{1\}\big\}$.
2. $\mathcal{Q} = \big\{\{1\}, \ldots, \{p\}\big\}$.
3. $\mathcal{Q} = \big\{\{1\}, \ldots, \{p\}, \{1,2\}, \ldots, \{p-1, p\}\big\}$.
4. $\mathcal{Q} = \big\{\{1\}, \{1,1\}, \ldots, \{\overbrace{1, \ldots, 1}^{d}\}\big\}$.

For the polynomial RKKS in the fourth example, one must also multiply the constants b_k to the λ's as appropriate. Let q be the cardinality of the set $\mathcal{Q}$, which is the number of summands required to construct the kernel for $\mathcal{F}$. Denote the instructed sets as $\xi = \{\xi_1, \ldots, \xi_q\}$ for λ and $a = \{a_1, \ldots, a_q\}$ for h. We can write the kernel h_λ as a linear combination of ξ and a,

$$h_\lambda = \xi_1 a_1 + \cdots + \xi_q a_q.$$

The reason this is important is because changes in λ for h_λ only changes the ξ_k's, but not the a_k's. This allows us to compute and store all of the required $n \times n$ kernel matrices $\mathbf{A}_1, \ldots, \mathbf{A}_q$ from the application of instruction set on h evaluated at all pairs of data points $(x_i, x_j) \in \mathcal{X} \times \mathcal{X}$. This process of initialisation need only be done once prior to commencing the EM algorithm—a step we refer to as 'kernel loading'. In the **iprior** package, kernel loading is performed using the `kernL()` command.

Notice that

$$\begin{aligned}
\operatorname{tr}\left(\boldsymbol{\Sigma}_\theta \tilde{\mathbf{W}}^{(t)}\right) &= \operatorname{tr}\left((\psi \mathbf{H}_\eta^2 + \psi^{-1}\mathbf{I}_n)\tilde{\mathbf{W}}^{(t)}\right) \\
&= \psi \operatorname{tr}(\mathbf{H}_\eta^2 \tilde{\mathbf{W}}^{(t)}) + \psi^{-1} \operatorname{tr} \tilde{\mathbf{W}}^{(t)} \\
&= \psi \operatorname{tr}\left(\sum_{j,k=1}^{q} \xi_j \xi_k \big(\mathbf{A}_j \mathbf{A}_k + (\mathbf{A}_j \mathbf{A}_k)^\top\big) \tilde{\mathbf{W}}^{(t)}\right) + \psi^{-1} \operatorname{tr} \tilde{\mathbf{W}}^{(t)} \\
&= 2\psi \sum_{j,k=1}^{q} \xi_j \xi_k \operatorname{tr}\left(\mathbf{A}_j \mathbf{A}_k \tilde{\mathbf{W}}^{(t)}\right) + \psi^{-1} \operatorname{tr} \tilde{\mathbf{W}}^{(t)}.
\end{aligned}$$

Provided that we have the matrices $\mathbf{A}_{jk} = \mathbf{A}_j \mathbf{A}_k$, $j, k = 1, \ldots, q$ in addition to $\mathbf{A}_1, \ldots, \mathbf{A}_q$ pre-calculated and stored, then evaluating $\operatorname{tr}\left(\mathbf{A}_{jk} \tilde{\mathbf{W}}^{(t)}\right) = \operatorname{vec}(\mathbf{A}_{jk})^\top \operatorname{vec}(\tilde{\mathbf{W}}^{(t)})$ is $O(n^2)$, although this only need to be done once per EM iteration. Thus, with the kernels loaded, the overall time complexity to evaluate Q is $O(n^2)$ at the beginning of each iteration, but roughly linear in ξ thereafter.

As a remark, we have achieved efficiency at the expense of storage and a potentially long initialisation phase of kernel loading. The storing of the kernel matrices a can be very expensive, especially if the sample size is very large. On the bright side, once the kernel matrices are stored in memory, the **iprior** package allows them to be reused again and again. A practical situation where this might be useful is when we would like to repeat the EM at various initial values.

4.3.3 The exponential family EM algorithm

In the original EM paper by Dempster et al. (1977), the EM algorithm was demonstrated to be easily administered to complete data likelihoods belonging to the exponential family for which the maximum likelihood estimates are easily computed. If this is the case, then the M-step simply involves replacing the unknown sufficient statistics in the ML estimates with their *conditional expectations* (see Section 4.9 for details). Certain I-prior models

emit this property, namely regression functions belonging to the full or limited ANOVA RKKS, and we describe its estimation below.

Assume A1–A3 applies, and that only the error precision ψ and the RKHS scale parameters $\lambda_1, \ldots, \lambda_p$ need to be estimated, i.e. all other kernel parameters are fixed—a similar situation was described in the previous subsection. For the full ANOVA RKKS, the kernel is

$$
\begin{aligned}
h_\lambda &= \sum_{i=1}^{p} \lambda_i h_i + \sum_{i<j} \lambda_i \lambda_j h_i h_j + \cdots + \prod_{i=1}^{p} \lambda_i h_i \\
&= \lambda_k \overbrace{\left(h_k + \sum_{i} \lambda_i h_i h_k + \cdots + h_k \prod_{i \neq k} \lambda_i h_i \right)}^{\text{terms of } \lambda_k} + \overbrace{\sum_{i \neq k} \lambda_i h_i + \sum_{i,j \neq k} \lambda_i \lambda_j h_i h_j + \cdots + 0}^{\text{no } \lambda_k \text{ here}} \\
&= \lambda_k r_k + s_k
\end{aligned}
$$

where r_k and s_k are both functions over $\mathcal{X} \times \mathcal{X}$, defined respectively as the terms of the ANOVA kernel involving λ_k, and the terms not involving λ_k. The reason for splitting h_λ like this will become apparently momentarily.

Programmatically this looks complicated to implement in software, but in fact it is not. Consider again the instruction list $\mathcal{Q}$ for the ANOVA RKKS (Example 3, Section 4.3.2). We can split this list into two: $\mathcal{R}_k$ as those elements of $\mathcal{Q}$ which involve the index k, and $\mathcal{S}_k$ as those elements of $\mathcal{Q}$ which do not involve the index k. Let ζ_k, e_k be the sets of λ and h after applying the instructions of $\mathcal{R}_k$, and let ξ_k and a_k be the sets of λ and h after applying the instructions of $\mathcal{S}_k$. Now, we have

$$
r_k = \frac{1}{\lambda_k} \sum_{i=1}^{|\mathcal{R}_k|} \zeta_{ik} e_{ik} \quad \text{and} \quad s_k = \sum_{i=1}^{|\mathcal{S}_k|} \xi_{ik} a_{ik}.
$$

Defining $\mathbf{R}_k$ and $\mathbf{S}_k$ as the kernel matrices with (i, j) entries $r_k(x_i, x_j)$ and $s_k(x_i, x_j)$ respectively, we have that

$$
\mathbf{H}_\eta^2 = \lambda_k^2 \mathbf{R}_k^2 + \lambda_k \overbrace{\left(\mathbf{R}_k \mathbf{S}_k + (\mathbf{R}_k \mathbf{S}_k)^\top \right)}^{\mathbf{U}_k} + \mathbf{S}_k^2.
$$

Consider now the full data log-likelihood for λ_k, $k = 1, \ldots, p$, conditionally dependent

on the rest of the unknown parameters ψ and $\lambda_{-k} = \{\lambda_1, \ldots, \lambda_p\} \backslash \{\lambda_k\}$:

$$
\begin{aligned}
L(\lambda_k | \mathbf{y}, \mathbf{w}, \lambda_{-k}, \psi) &= \text{const.} - \frac{1}{2} \operatorname{tr}\left((\psi \mathbf{H}_\eta^2 + \psi^{-1} \mathbf{I}_n) \mathbf{w}\mathbf{w}^\top \right) + \psi \tilde{\mathbf{y}}^\top \mathbf{H}_\eta \mathbf{w} \\
&= \text{const.} - \lambda_k^2 \cdot \frac{\psi}{2} \operatorname{tr}(\mathbf{R}_k^2 \mathbf{w}\mathbf{w}^\top) + \lambda_k \cdot \left(\psi \tilde{\mathbf{y}}^\top \mathbf{R}_k \mathbf{w} - \frac{\psi}{2} \operatorname{tr}(\mathbf{U}_k \mathbf{w}\mathbf{w}^\top) \right).
\end{aligned} \tag{4.14}
$$

{eq:loglikl ambdak}

Notice that the above likelihood is an exponential family distribution with the natural parameterisation $\beta = (-\lambda_k^2, \lambda_k)$ and sufficient statistics T_1 and T_2 defined by

$$
T_1 = \frac{\psi}{2} \operatorname{tr}(\mathbf{R}_k^2 \mathbf{w}\mathbf{w}^\top) \quad \text{and} \quad T_2 = \psi \tilde{\mathbf{y}}^\top \mathbf{R}_k \mathbf{w} - \frac{\psi}{2} \operatorname{tr}(\mathbf{U}_k^2 \mathbf{w}\mathbf{w}^\top).
$$

This likelihood is maximised at $\hat{\lambda}_k = T_2 / 2T_1$, but of course, the variables $w_1, \ldots, w_n$ are never observed. As per the exponential family EM routine, replace occurrences of $\mathbf{w}$ and $\mathbf{w}\mathbf{w}^\top$ with their respective conditional expectations, i.e. $\mathbf{w} \mapsto \mathrm{E}[\mathbf{w}|\mathbf{y}] = \tilde{\mathbf{w}}$ and $\mathbf{w}\mathbf{w}^\top \mapsto \mathrm{E}[\mathbf{w}\mathbf{w}^\top|\mathbf{y}] = \tilde{\mathbf{V}}_w + \tilde{\mathbf{w}}\tilde{\mathbf{w}}^\top$ as defined in (4.7). That the λ_k's have closed-form expressions, together with the closed-form expression for ψ in (4.12), greatly simplifies the EM algorithm. At the M-step, one simply updates the parameters in turn, and as such, there is no maximisation per se.

The algorithm is summarised in Algorithm 1. The exponential family EM for ANOVA-type I-prior models require $O(n^3)$ computational time at each step, which is spent on computing the matrix inverse in the E-step. The M-step takes at most $O(n^2)$ time to compute. As a remark, it is not necessary that h_λ is the full ANOVA RKKS; any of the examples 1–3 in Section 4.3.2 can be estimated using this method, since they are seen as special cases of the ANOVA decomposition.

While the exponential family EM algorithm takes similar computational time as the efficient EM algorithm described in Section 4.3.2, there is one compelling reason to consider Algorithm 1: conjugacy of the exponential family of distributions. Realise that $\lambda_k | (\mathbf{y}, \mathbf{w}, \lambda_{-k}, \psi)$ is in fact normally distributed, with mean and variance given by $T_2/2T_1$ and $1/2T_1$ respectively. If we were so compelled to assign a normal prior on each of the λ_k's, then the conditionally dependent log-likelihood of λ_k, $L(\lambda_k | \mathbf{y}, \mathbf{w}, \lambda_{-k}, \psi)$, would have a normal log-likelihood prior involving λ_k added on. Importantly, viewed as a posterior log-density for λ_k, the posterior density for λ_k would also be a normal distribution. The EM as a whole would then generate maximum a posteriori (MAP) estimates for the parameters. Although not shown here, similar conjugacy benefits for the ψ parameter can be argues, whereby the gamma distribution is the density in question. The usual

alg:EM2

Algorithm 1 Exponential family EM for ANOVA-type I-prior models

procedure INITIALISATION
  Initialise $\lambda_1^{(0)}, \ldots, \lambda_p^{(0)}, \psi^{(0)}$
  Compute and store matrices as per $\mathcal{R}_k$ and $\mathcal{S}_k$.
  $t \leftarrow 0$
end procedure

while not converged **do**
  procedure E-STEP
    $\tilde{\mathbf{w}} \leftarrow \psi^{(t)} \mathbf{H}_{\eta^{(t)}} \left(\psi^{(t)} \mathbf{H}_{\eta^{(t)}}^2 + \psi^{-(t)} \mathbf{I}_n \right)^{-1} \tilde{\mathbf{y}}$
    $\tilde{\mathbf{W}} \leftarrow \left(\psi^{(t)} \mathbf{H}_{\eta^{(t)}}^2 + \psi^{-(t)} \mathbf{I}_n \right)^{-1} + \tilde{\mathbf{w}} \tilde{\mathbf{w}}^\top$
  end procedure

  procedure M-STEP
    for $k = 1, \ldots, p$ **do**
      $T_{1k} \leftarrow \frac{1}{2} \operatorname{tr}(\mathbf{R}_k^2 \tilde{\mathbf{W}})$
      $T_{2k} \leftarrow \tilde{\mathbf{y}}^\top \mathbf{R}_k \tilde{\mathbf{w}} - \frac{1}{2} \operatorname{tr}(\mathbf{U}_k^2 \tilde{\mathbf{W}}^\top)$
      $\lambda_k^{(t+1)} \leftarrow T_{2k} / 2 T_{1k}$
    end for
    $T_3 \leftarrow \tilde{\mathbf{y}}^\top \tilde{\mathbf{y}} + \operatorname{tr}(\mathbf{H}_{\eta^{(t)}}^2 \tilde{\mathbf{W}}^{(t)}) - 2 \tilde{\mathbf{y}}^\top \mathbf{H}_{\eta^{(t)}} \tilde{\mathbf{w}}^{(t)}$
    $\psi^{(t+1)} \leftarrow \operatorname{tr} \tilde{\mathbf{W}}^{(t)} / T_3$
  end procedure
  $t \leftarrow t + 1$
end while

EM algorithm without using any priors can be viewed as using improper priors for the parameters, i.e. $p(\lambda_k) \propto$ const. and $p(\psi) \propto$ const..

In the next chapter on binary and multinomial regression using I-priors, the exponential family EM algorithm described here is especially relevant, as it is connected to the variational Bayesian algorithm (Bernardo et al., 2003) that will be used for estimating the models described therein.

Remark 4.5. Earlier, we restricted attention to ANOVA RKKS. Hopefully, it is now apparent that ANOVA kernels are a requirement for Algorithm 1 to work easily. As soon as higher degrees of the λ_k's come into play, e.g. using the polynomial kernel, then the ML estimate for λ_k involve solving a polynomial of degree $2d - 1$ the FOC equations. Although this is not in itself hard to do, the elegance of the algorithm, especially viewed as having the normal conjugacy property for the $\lambda_k's$, is lost.

4.3.4 Accelerating the EM algorithm

A criticism of the EM algorithm is that it may take many iterations to converge. Several novel ideas have been looked at in a bid to 'accelerate the EM algorithm', as it were. One such approach, which does not require any amendment to the particular EM algorithm at hand, is called the *monotonically over-relaxed EM algorithm* (MOEM) by Yu (2012).

The idea of MOEM is as follows. At every iteration of the MOEM, perform as usual the E-step and M-step to obtain an updated parameter value $\theta_{\text{EM}}^{(t+1)}$. Instead of using this update value of the parameter, modify it instead, and use

$$\theta^{(t+1)} = (1+\omega)\theta_{\text{EM}}^{(t+1)} - \omega\theta^{(t)},$$

where ω is an *over-relaxation* parameter. Under mild conditions, among them that $Q(\theta^{(t+1)}) > Q(\theta^{(t)})$, the MOEM estimate does not decrease the log-likelihood at each step. This condition is a slight inconvenience to check under the usual EM algorithm, but is a great companion to exponential family EM algorithm. From (4.14), we see that $Q(\lambda_k) = \mathrm{E}_{\mathbf{w}}\left[L(\lambda_k|\theta\backslash\{\lambda_k\})|\mathbf{y}, \theta^{(t)}\right]$ is quadratic in λ_k, therefore any $\omega \in [0, 1]$ will maintain monotonicity of the EM algorithm.

4.4 Post-estimation

One of the perks of a (semi-)Bayesian approach to regression modelling is that we are able to use Bayesian post-estimation machinery involving the relevant posterior distributions. With the normal I-prior model, there is the added benefit that posterior distributions are easily obtained in closed form. The plots that are shown in this subsection is a continuation of the example from subsection 4.2.5.

Recall that for the I-prior model (4.6), the regression function $f(x) = \sum_{i=1}^{n} h_{\hat{\eta}}(x, x_i)\tilde{w}_i$ has the posterior Gaussian distribution specified by the multivariate-normal mean and variance of the $\tilde{w}_i$'s given in (4.7). Denote by $\mathbf{h}_{\hat{\eta}}(x)$ the n-vector with entries equal to $h_{\hat{\eta}}(x, x_i)$. Precisely, the posterior density for the regression function is

$$p\big(f(x)|\mathbf{y}\big) \sim \mathrm{N}\left(\mathbf{h}_{\hat{\eta}}(x)\hat{\mathbf{w}}, \mathbf{h}_{\hat{\eta}}(x)^{\top}\big(\mathbf{H}_{\hat{\eta}}\hat{\mathbf{\Psi}}\mathbf{H}_{\hat{\eta}} + \hat{\mathbf{\Psi}}^{-1}\big)^{-1}\mathbf{h}_{\hat{\eta}}(x)\right) \tag{4.15}$$

for any x in the domain of the regression function. Here, the hats on the parameters indicate the use of the optimised model parameters, i.e. the ML or MAP estimates.

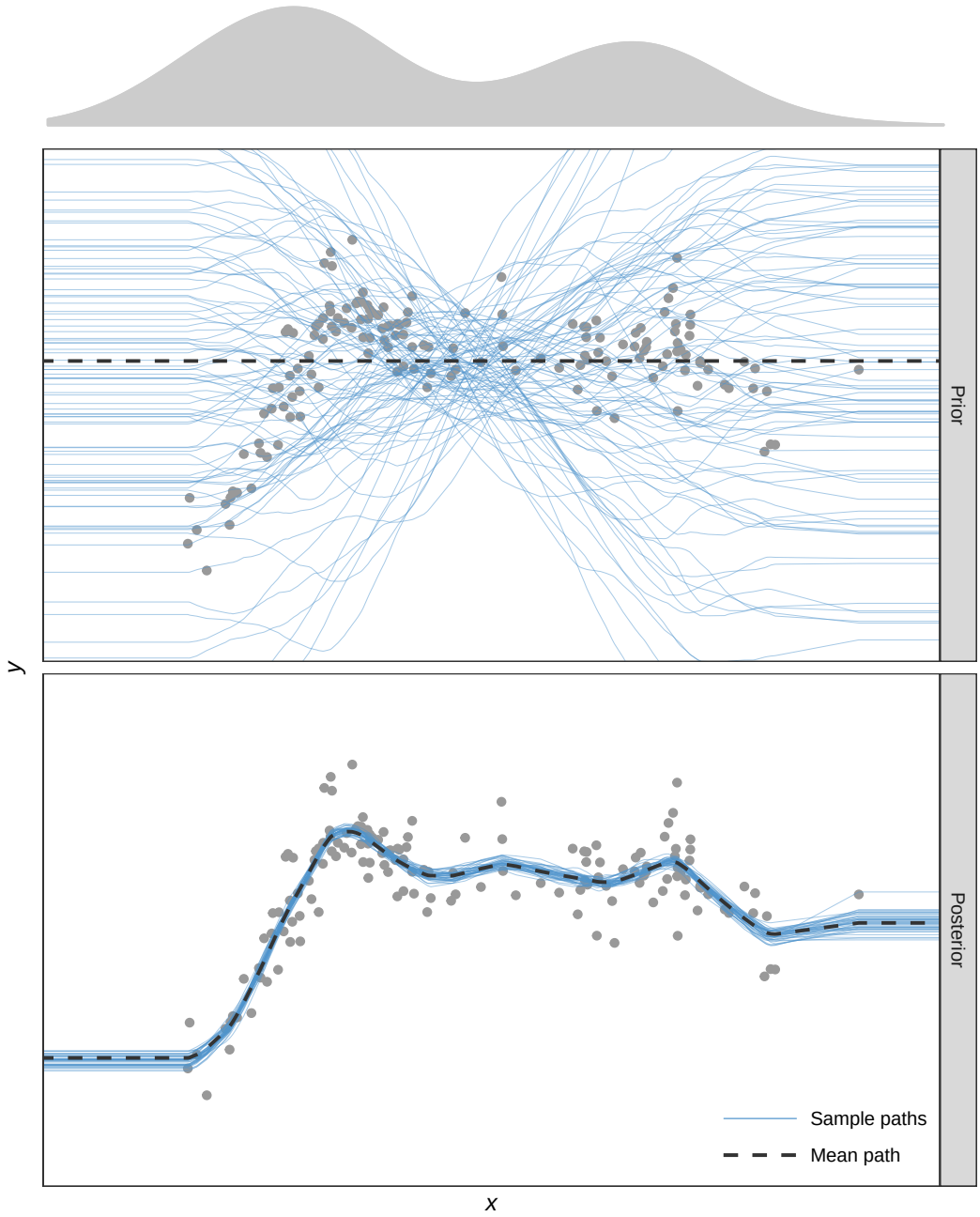


Figure 4.2: Prior (top) and posterior (bottom) sample path realisations of regression functions drawn from their respective distributions when $\mathcal{F}$ is a fBm-0.5 RKHS. At the very top of the figure, a smoothed density estimate of the x's is overlaid. In regions with few data points (near the centre), there is little Fisher information, and hence a conservative prior closer to zero, the prior mean, for this region.

Prediction of a new data point is also of interest. A priori, assume that $y_{\text{new}} = \hat{\alpha} + f(x_{\text{new}}) + \epsilon_{\text{new}}$, where $\epsilon_{\text{new}} \sim \text{N}(0, \psi_{\text{new}}^{-1})$, and $f \sim$ I-prior. Denote the covariance between ϵ_{new} and $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)^\top$ by $\boldsymbol{\sigma}_{\text{new}}^\top \in \mathbb{R}^n$. Under an iid model (assumption A3), then $\psi_{\text{new}} = \psi = \text{Var}\, \epsilon_i$ for any $i \in \{1, \ldots, n\}$, and $\boldsymbol{\sigma}_{\text{new}}^\top = \mathbf{0}$, but otherwise, these extra parameters need to be dealt with somehow, either by specifying them a priori or estimating them again, which seems excessive. In any case, using a linearity argument, the posterior distribution for y_{new} is normal, with mean and variance given by

$$\text{E}[y_{\text{new}}|\mathbf{y}] = \hat{\alpha} + \text{E}\left[f(x_{\text{new}})|\mathbf{y}\right] + \text{correction term} \tag{4.16}$$

and

$$\text{Var}[y_{\text{new}}|\mathbf{y}] = \text{Var}\left[f(x_{\text{new}})|\mathbf{y}\right] + \psi_{\text{new}}^{-1} + \text{correction term}. \tag{4.17}$$

A derivation is presented in section 4.10. Note, that the mean and variance correction term vanishes under an iid assumption A3. The posterior distribution for y_{new} can be used in several ways. Among them, is to construct a $100(1-\alpha)\%$ credibility interval for the (mean) predicted value y_{new} using

$$\text{E}[y_{\text{new}}|\mathbf{y}] \pm \Phi^{-1}(1-\alpha/2) \cdot \text{Var}[y_{\text{new}}|\mathbf{y}]^{\frac{1}{2}},$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function. One could also perform a posterior predictive density check of the data $\mathbf{y}$, by repeatedly sampling n points from its posterior distribution. This provides a visual check of whether there are any systematic deviances between what the model predicts, and what is observed from the data.

Lastly, we discuss model comparison. Recall that the marginal distribution for $\mathbf{y}$ after integrating out the I-prior for f in model (4.6) is a normal distribution. Suppose that we are interested in comparing two candidate models M_1 and M_2, each with the parameter set θ_1 and θ_2. Commonly, we would like to test whether or not particular terms in the ANOVA RKKS are significant contributors in explaining the relationship between the responses and predictors. A log-likelihood comparison is possible using an asymptotic chi-squared distribution, with degrees of freedom equal to the difference between the number of parameters in θ_2 and θ_1. This is assuming model M_1 is nested within M_2, which is the case for ANOVA-type constructions. Note that if two models have the same number of parameters, then the model with the higher likelihood is preferred.

Remark 4.6. This method of comparing marginal likelihoods can be seen as Bayesian

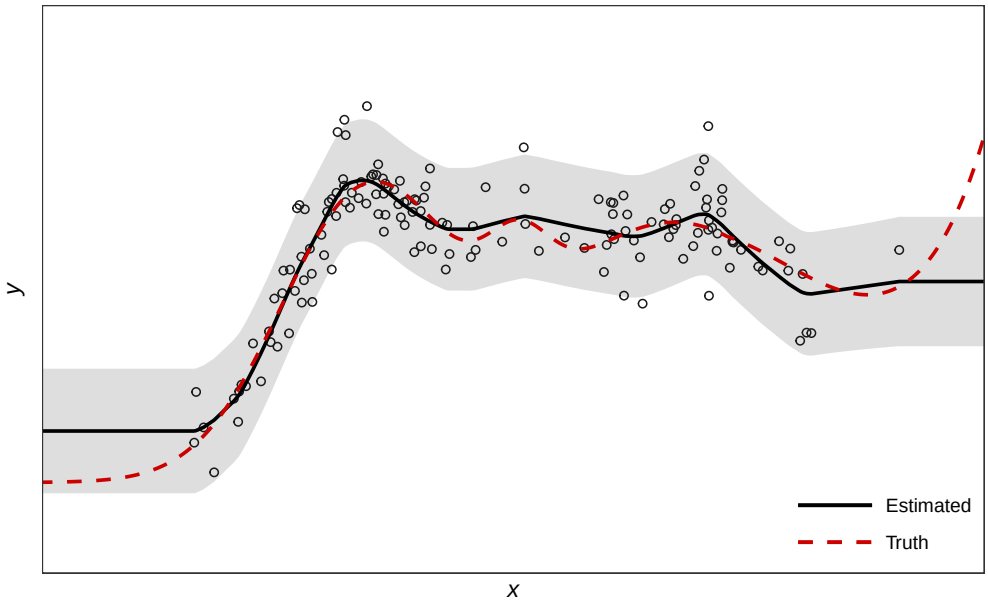


Figure 4.3: The estimated regression line (solid black) is the posterior mean estimate of the regression function (shifted by the intercept), which also gives the posterior mean estimate for the responses y. The shaded region is the 95% credibility interval for predictions. The true regression line (dashed red) is shown for comparison.

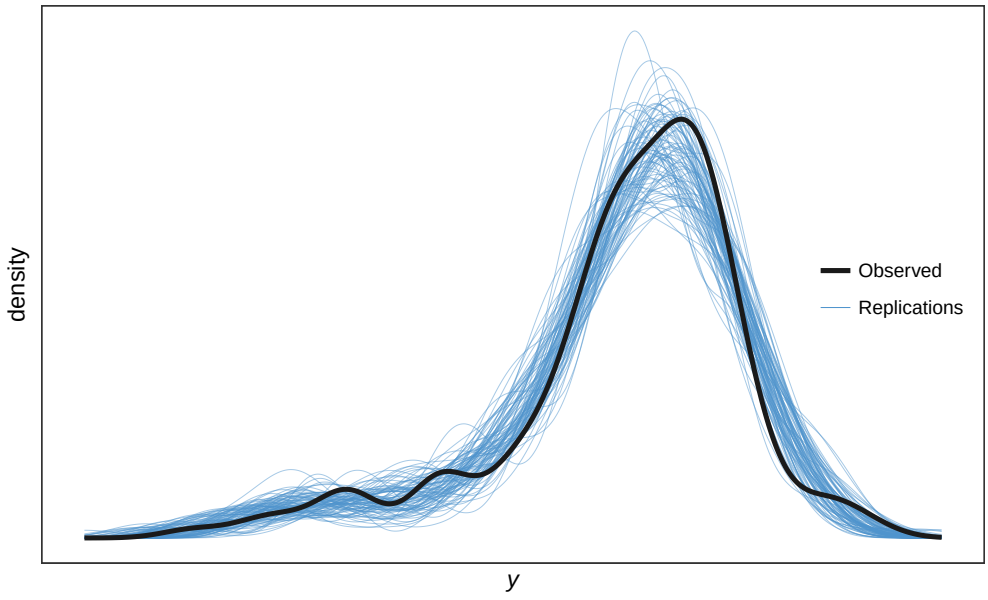


Figure 4.4: Posterior predictive density checks of the responses: repeated sampling from the posterior density of the y_i's and plotting their densities allows us to compare model predictions against observed samples.

model selection using *empirical Bayes factors*, where the Bayes factor of comparing model M_1 to model M_2 is defined as

$$\text{BF}(M_1, M_2) = \frac{\int p(\mathbf{y}|\theta_1, \mathbf{f})p(\mathbf{f})\,\mathrm{d}\mathbf{f}}{\int p(\mathbf{y}|\theta_2, \mathbf{f})p(\mathbf{f})\,\mathrm{d}\mathbf{f}}.$$

The word 'empirical' stems from the fact that the parameters are estimated via an empirical Bayes approach (maximum marginal likelihood). This approach is fine when the number of comparisons to be made is small, but can be computationally unfeasible when many marginal likelihoods need to be pairwise compared. In Chapter 6, we explore a fully Bayesian approach to explore the entire model space for the special case of linear models.

4.5 Examples

sec:ipriorexamples

We demonstrate I-prior modelling on a toy data set to illustrate the Nyström method, as well as three other real-data examples. All of the analyses were conducted in R, and I-prior model estimation was done using the **iprior** package. In all of these examples, A1–A3 were assumed.

4.5.1 Using the Nyström method

We investigate the use of the Nyström method of approximating the kernel matrix in estimating I-prior models. Let us revisit the data set generated by (4.13) described in Section 4.2.5. The features of this regression function are two large bumps at the centres of the mixed Gaussian PDFs, and also a small bump right after $x > 4.5$ caused by the additional exponential function. The true regression function goes to positive infinity as x increases, and to zero as x decreases. Samples of (x_i, y_i), $i = 1, \ldots, 2000$ have been generated by the built-in `gen_smooth()` function, of which the first few lines of the data are shown below.

```
R> dat <- gen_smooth(n = 2000, xlim = c(-1, 5.5), seed = 1)
R> head(dat)
```

```
##           y         X
## 1 0.6803514 -2.608953
```

```
## 2  3.6747031 -2.554039
## 3 -1.1563508 -2.381275
## 4  2.2657657 -2.280259
## 5  2.5398243 -2.214122
## 6  1.2929592 -2.170532
```

One could fit the regression model using all available data points, with an I-prior from the fBm-0.5 RKHS of functions as follows (note that the `silent` option is used to suppress the output from the `iprior()` function):

```
R> (mod.full <- iprior(y ~ X, dat, kernel = "fbm",
+                       control = list(silent = TRUE)))
```

```
## Log-likelihood value: -4355.075
##
##  lambda     psi
## 2.30244 0.23306
```

To implement the Nyström method, the option `nystrom = 50` was added to the above function call, which uses 50 randomly selected data points for the Nyström approximation.

```
R> (mod.nys <- iprior(y ~ X, dat, kernel = "fbm", nystrom = 50,
+                      control = list(silent = TRUE)))
```

```
## Log-likelihood value: -1945.33
##
##  lambda     psi
## 1.64833 0.13538
```

The hyperparameters estimated for both models are slightly different. The log-likelihood is also different, but this is attributed to information loss due to the approximation procedure. Nevertheless, we see from Figure 4.5 that the estimated regression functions are quite similar in both the full model and the approximated model. The main difference is that the the Nyström method was not able to extrapolate the right hand side of the plot well, because it turns out that there were no data points used from this region. This can certainly be improved by using a more intelligent sampling scheme. The full model took a little under 15 minutes to converge, while the Nyström method took just seconds. Storage savings is significantly higher with the Nyström method as

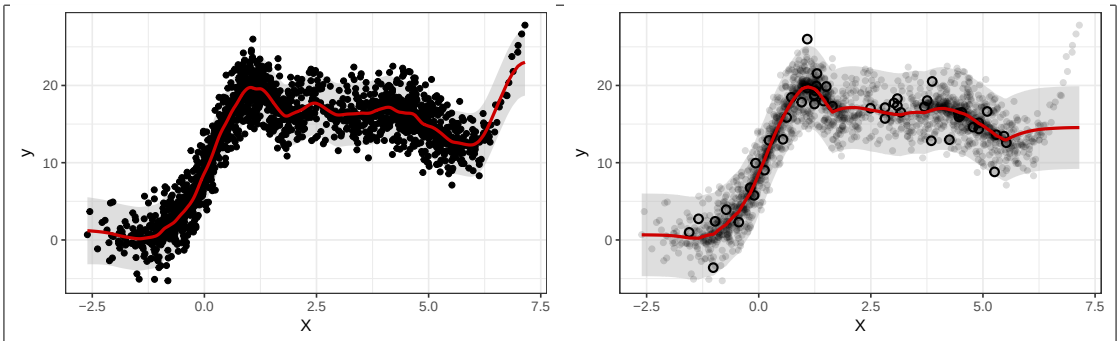


Figure 4.5: Plot of predicted regression function for the full model (left) and the Nyström approximated method (right). For the Nyström plot, the data points that were active are shown by circles with bold outlines.

fig:nystrom.plot

well.

```
R> get_time(mod.full); get_size(mod.full, units = "MB")

## 14.63474 mins
## 128.2 MB

R> get_time(mod.nys); get_size(mod.nys)

## 1.324355 secs
## 965.2 kB
```

4.5.2 Random effects models

In this section, a comparison between a standard random effects model and the I-prior approach for estimating varying intercept and slopes model is illustrated. The example concerns control data[3] from several runs of radioimmunoassays (RIA) for the protein insulin-like growth factor (IGF-I) (explained in further detail in Davidian and Giltinan, 1995, §3.2.1). RIA is a in vitro assay technique which is used to measure concentration of antigens—in our case, the IGF-I proteins. When an RIA is run, control samples at known concentrations obtained from a particular lot are included for the purpose of assay quality control. It is expected that the concentration of the control material remains stable as the machine is used, up to a maximum of about 50 days, at which point control samples from a new batch is used to avoid degradation in assay performance.

```
R> data(IGF, package = "nlme")
R> head(IGF)
```

```
## Grouped Data: conc ~ age | Lot
##   Lot age conc
## 1   1   7 4.90
## 2   1   7 5.68
## 3   1   8 5.32
## 4   1   8 5.50
## 5   1  13 4.94
## 6   1  13 5.19
```

The data consists of IGF-I concentrations (`conc`) from control samples from 10 different lots measured at differing `age`s of the lot. The data were collected with the aim of identifying possible trends in control values `conc` with `age`, ultimately investigating whether or not the usage protocol of maximum sample age of 50 days is justified. J. C. Pinheiro and Bates (2000) remarks that this is not considered a longitudinal problem because different samples were used at each measurement.

We shall model the IGF data set using the I-prior methodology using the ANOVA-decomposed regression function

$$f(\texttt{age}, \texttt{Lot}) = f_1(\texttt{age}) + f_2(\texttt{Lot}) + f_{12}(\texttt{age}, \texttt{Lot})$$

where f_1 lies in the linear RKHS $\mathcal{F}_1$, f_2 in the Pearson RKHS $\mathcal{F}_2$ and f_{12} in the tensor product space $\mathcal{F}_{12} = \mathcal{F}_1 \otimes \mathcal{F}_2$. The regression function f then lies in the RKHS $\mathcal{F} = \mathcal{F}_1 \oplus \mathcal{F}_2 \oplus \mathcal{F}_{12}$ with kernel equal to the sum of the kernels from each of the RKHSs. The explanation here is that the `conc` levels are assumed to be related to both `age` and `Lot`, and in particular, the contribution of `age` on `conc` varies with each individual `Lot`. This gives the intended effect of a linear mixed-effects model, which is thought to be suitable in this case, in order to account for within-lot and between-lot variability. We first fit the model using the **iprior** package, and then compare the results with the standard random effects model using `lme4::lmer()`. The command to fit the I-prior model using the EM algorithm is

```
R> mod.iprior <- iprior(conc ~ age * Lot, IGF, method = "em")
```

[3] This data is available in the R package **nlme** (J. Pinheiro et al., 2017).

```
## ========================================
## Converged after 57 iterations.

R> summary(mod.iprior)

## Call:
## iprior(formula = conc ~ age * Lot, data = IGF, method = "em")
##
## RKHS used:
## Linear (age)
## Pearson (Lot)
##
## Residuals:
##    Min. 1st Qu.  Median 3rd Qu.    Max.
## -4.4889 -0.3798 -0.0090  0.2563  4.3973
##
## Hyperparameters:
##           Estimate   S.E.      z P[|Z>z|]
## lambda[1]   0.0000 0.0002 -0.004    0.997
## lambda[2]   0.0007 0.0030  0.238    0.812
## psi         1.4576 0.1366 10.672   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Closed-form EM algorithm. Iterations: 57/100
## Converged to within 1e-08 tolerance. Time taken: 3.043089 secs
## Log-likelihood value: -291.9033
## RMSE of prediction: 0.8273639 (Training)
```

To make inference on the covariates, we look at the scale parameters `lambda`. We see that both scale parameters for `age` and `Lot` are close to zero, and a test of significance is not able to reject the hypothesis that these parameters are indeed null. We conclude that neither `age` nor `Lot` has a linear effect on the `conc` levels. The plot of the fitted regression line in Figure 4.6 does show an almost horizontal line for each `Lot`.

The standard random effects model, as explored by Davidian and Giltinan (1995) and

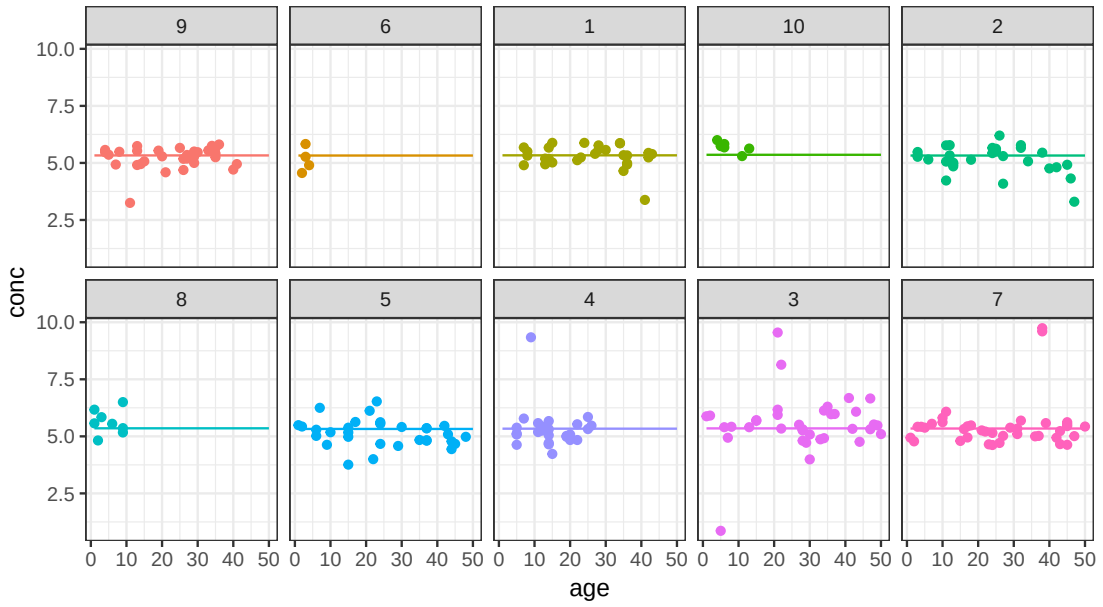


Figure 4.6: Plot of fitted regression line for the I-prior model on the IGF data set, separated into each of the 10 lots.

fig:IGF.mod.iprior.plot

J. C. Pinheiro and Bates (2000), is

$$\texttt{conc}_{ij} = \beta_{0j} + \beta_{1j}\texttt{age}_{ij} + \epsilon_{ij}$$
$$\begin{pmatrix}\beta_{0j}\\ \beta_{1j}\end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix}, \begin{pmatrix}\sigma_0^2 & \sigma_{01}\\ \sigma_{01} & \sigma_1^2\end{pmatrix}\right)$$
$$\epsilon_{ij} \sim \mathrm{N}(0, \sigma^2)$$

for $i = 1, \ldots, n_j$ and the index j representing the 10 `Lots`. Fitting this model using `lmer`, we can test for the significance of the fixed effect β_0, for which we find that it is not (p-value $= 0.616$), and arrive at the same conclusion as in the I-prior model. However, we notice that the package reports a perfect negative correlation between the random effects, σ_{01}. This indicates a potential numerical issue when fitting the model—a value of exactly -1, 0 or 1 is typically imposed by the package to force through estimation in the event of non-positive definite covariance matrices arising. We can inspect the eigenvalues of the covariance matrix for the random effects to check that they are indeed non-positive definite.

```
R> (mod.lmer <- lmer(conc ~ age + (age | Lot), IGF))

## Linear mixed model fit by REML ['lmerMod']
## Formula: conc ~ age + (age | Lot)
##    Data: IGF
## REML criterion at convergence: 594.3662
## Random effects:
##  Groups   Name        Std.Dev. Corr
##  Lot      (Intercept) 0.082507
##           age         0.008092 -1.00
##  Residual             0.820628
## Number of obs: 237, groups:  Lot, 10
## Fixed Effects:
## (Intercept)          age
##    5.374974    -0.002535

R> eigen(VarCorr(mod.lmer)$Lot)

## eigen() decomposition
## $values
## [1]  6.872939e-03 -1.355253e-20
##
## $vectors
##             [,1]        [,2]
## [1,] -0.99522490 -0.09760839
## [2,]  0.09760839 -0.99522490
```

Degenerate covariance matrices often occur in models with a large number of random coefficients. These are typically solved by setting restrictions which then avoids overparameterising the model. One advantage of the I-prior method for varying intercept/slopes model is that the positive-definiteness is automatically taken care of. Furthermore, I-prior models typically require less number of parameters to fit a similar varying intercept/slopes

Table 4.2: A comparison of the estimates for the covariance matrix of the random effects using the I-prior model and the standard random effects model.

tab:igf

Parameter	iprior	lmer
σ_0	0.012	0.083
σ_1	0.000	0.008
ρ_{01}	0.690	-1.000

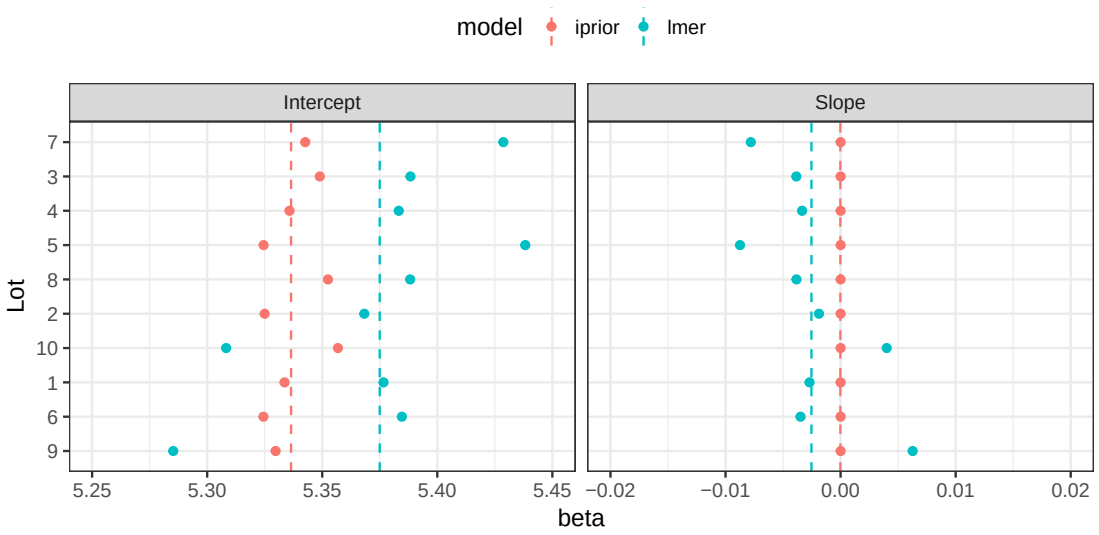


Figure 4.7: A comparison of the estimates for random intercepts and slopes (denoted as points) using the I-prior model and the standard random effects model. The dashed vertical lines indicate the fixed effect values.

fig:IGF.plot.beta

model – in the above example, the I-prior model estimated only three parameters, while the standard random effects model estimated a total of six parameters.

It is also possible to "recover" the estimates of the standard random effects model from the I-prior model, albeit in a slighly manual fashion (refer to subsection 4.1.2). Denote by f^j the individual linear regression lines for each of the $j = 1, \ldots, 10$ `Lots`. Then, each of these f^j has a slope and intercept for which we can estimate from the fitted values $\hat{f}^j(x_{ij})$, $i = 1, \ldots, n_j$. This would give us the estimate of the posterior mean of the random intercepts and slopes; these would typically be obtained using empirical-Bayes methods in the case of the standard random effects model.

Furthermore, σ_0^2 and σ_1^2 gives a measure of variability of the intercepts and slopes of the different groups, and this can be calculated from the estimates of the random intercepts and slopes. In the same spirit, $\rho_{01} = \sigma_{01}/(\sigma_0\sigma_1)$, which is the correlation between the random intercept and slope, can be similarly calculated. Finally, the fixed effects can be estimated from the intercept and slope of the best fit line running through the I-prior estimated `conc` values. The intuition for this is that the fixed effects are essentially the ordinary least squares (OLS) of a linear model if the groupings are disregarded. Figure 4.7 illustrates the differences in the estimates for the random coefficients, while Table 4.2 illustrates the differences in the estimates for the covariance matrix. Minor dif-

ferences do exist, with the most noticeable one being that the slopes in the I-prior model are categorically estimated as zero, and the sign of the correlation ρ_{01} being opposite in both models. Even so, the conclusions from both models are similar.

4.5.3 Longitudinal data analysis

sec:cows

We consider a balanced longitudinal data set consisting of weights in kilograms of 60 cows, 30 of which were randomly assigned to treatment group A, and the remaining 30 to treatment group B. The animals were weighed 11 times over a 133-day period; the first 10 measurements for each animal were made at two-week intervals and the last measurement was made one week later. This experiment was reported by Kenward (1987), and the data set is included as part of the package **jmcm** (J. Pan and Y. Pan, 2016) in R. The variable names have been renamed for convenience.

```
R> data(cattle, package = "jmcm")
R> names(cattle) <- c("id", "time", "group", "weight")
R> cattle$id <- as.factor(cattle$id)  # convert to factors
R> str(cattle)
```

```
## 'data.frame': 660 obs. of  4 variables:
##  $ id    : Factor w/ 60 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1..
##  $ time  : num  0 14 28 42 56 70 84 98 112 126 ...
##  $ group : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 1 ...
##  $ weight: int  233 224 245 258 271 287 287 287 290 293 ...
```

The response variable of interest are the `weight` growth curves, and the aim is to investigate whether a treatment effect is present. The usual approach to analyse a longitudinal data set such as this one is to assume that the observed growth curves are realizations of a Gaussian process. For example, Kenward (1987) assumed a so-called ante-dependence structure of order k, which assumes an observation depends on the previous k observations, but given these, is independent of any preceeding observations.

Using the I-prior, it is not necessary to assume the growth curves were drawn randomly. Instead, it suffices to assume that they lie in an appropriate function class. For this example, we assume that the function class is the fBm RKHS, i.e., we assume a smooth effect of time on weight. The growth curves form a multidimensional (or functional) response equivalent to a "wide" format of representing repeated measures data. In our analysis using the **iprior** package, we used the "long" format and thus our (uni-

tab:cowmodel

Table 4.3: A brief description of the five models fitted using I-priors.

Model	Explanation	Formula (`weight ~ ...`)
1	Growth does not vary with treatment nor among cows	`time`
2	Growth varies among cows only	`id * time`
3	Growth varies with treatment only	`group * time`
4	Growth varies with treatment and among cows	`id * time + group * time`
5	Growth varies with treatment and among cows, with an interaction effect between treatment and cows	`id * group * time`

dimensional) sample size n is equal to 60 cows $\times$ 11 repeated measurements. We also have two covariates potentially influencing growth, namely the cow subject `id` and also treatment `group`. The regression model can then be thought of as

$$\texttt{weight} = \alpha + f(\texttt{id}, \texttt{group}, \texttt{time}) + \epsilon$$
$$\epsilon \sim \mathrm{N}(0, \psi^{-1}).$$

We assume iid errors, and in addition to a smooth effect of `time`, we further assume a nominal effect of both cow `id` and treatment `group` using the Pearson RKHS. In the **iprior** package, factor type objects are treated with the Pearson kernel automatically, and the only `model` option we need to specify is the `kernel = "fbm"` option for the `time` variable. We have opted not to estimate the Hurst coefficient in the interest of computational time, and instead left it at the default value of 0.5. Table 4.3 explains the five models we have fitted.

The simplest model fitted was one in which the growth curves do not depend on the treatment effect or individual cows. We then added treatment effect and the cow `id` as covariates, separately first and then together at once. We also assumed that both of these covariates are time-varying, and hence added also the interaction between these covariates and the `time` variable. The final model was one in which an interaction between treatment effect and individual cows was assumed, which varied over time.

All models were fitted using the `mixed` estimation method. Compared to the EM algorithm alone, we found that the combination of direct optimisation with the EM algorithm in the `mixed` routine fits the model about six times faster for this data set due

tab:cowresults

Table 4.4: Summary of the five I-prior models fitted to the cow data set.

Model	Formula (`weight ~ ...`)	Log-likelihood	Error S.D.	Number of parameters
1	`time`	-2789.23	16.33	1
2	`id * time`	-2789.60	16.35	2
3	`group * time`	-2295.16	3.68	2
4	`id * time + group * time`	-2270.85	3.39	3
5	`id * group * time`	-2249.26	3.90	3

to slow convergence of EM algorithm. Here is the code and output for fitting the first model:

```
R> # Model 1: weight ~ f(time)
R> set.seed(456)
R> (mod1 <- iprior(weight ~ time, cattle, kernel = "fbm", method = "mixed"))
## Running 5 initial EM iterations
## ================================================================================
## Now switching to direct optimisation
## final  value 1394.615062
## converged
## Log-likelihood value: -2789.231
##
##  lambda     psi
## 0.83592 0.00375
```

The results of the model fit are summarised in Table 4.4. We can test for a treatment effect by testing Model 4 against the alternative that Model 2 is true. The log-likelihood ratio test statistic is $D = -2(-2789.60-(-2270.85)) = 1037.49$ which has an asymptotic chi-squared distribution with $3-2=1$ degree of freedom. The p-value for this likelihood ratio test is less than 10^{-6}, so we conclude that Model 4 is significantly better than Model 2.

We can next investigate whether the treatment effect differs among cows by comparing Model 5 against Model 4. As these models have the same number of parameters, we can simply choose the one with the higher likelihood, which is Model 5. We conclude that treatment does indeed have an effect on growth, and that the treatment effect differs

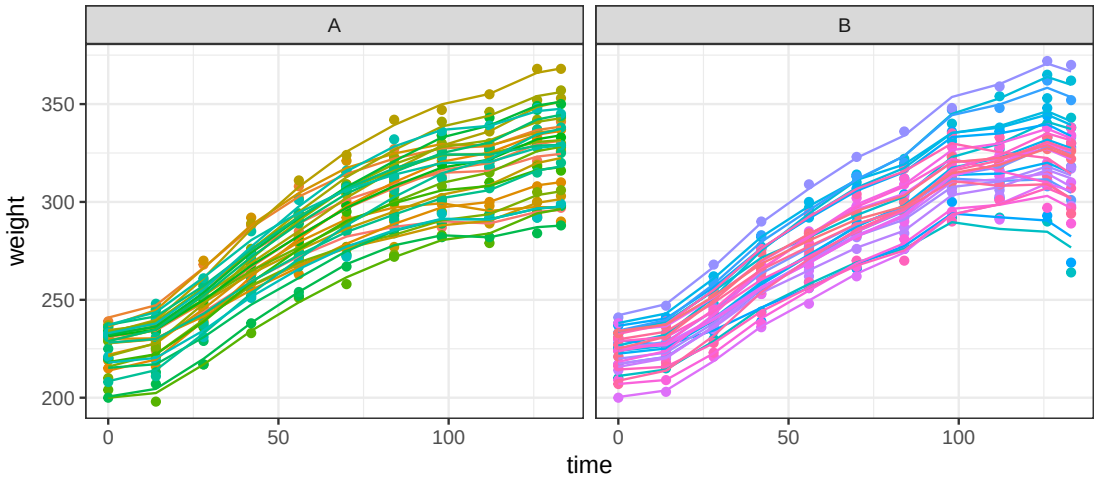


Figure 4.8: A plot of the I-prior fitted regression curves from Model 5. In this model, growth curves differ among cows and by treatment effect (with an interaction between cows and treatment effect), thus producing these 60 individual lines, one for each cow, split between their respective treatment groups (A or B).

fig:cows.plot

among cows. A plot of the fitted regression curves onto the cow data set is shown in Figure 4.8.

4.5.4 Regression with a functional covariate

We illustrate the prediction of a real valued response with a functional covariate using a widely analysed data set for quality control in the food industry. The data[4] contain samples of spectrometric curve of absorbances of 215 pieces of finely chopped meat, along with their water, fat and protein content. These data are recorded on a Tecator Infratec Food and Feed Analyzer working in the wavelength range 850–1050 nm by the Near Infrared Transmission (NIT) principle. Absorption data has not been measured continuously, but instead 100 distinct wavelengths were obtained. Figure 4.9 shows a sample of 10 such spectrometric curves.

For our analyses and many others' in the literature, the first 172 observations in the data set are used as a training sample for model fitting, and the remaining 43 observations as a test sample to evaluate the predictive performance of the fitted model. The

[4] Obtained from Tecator (see `http://lib.stat.cmu.edu/datasets/tecator` for details). We used the version made available in the dataframe `tecator` from the R package **caret** (Kuhn et al., 2017).

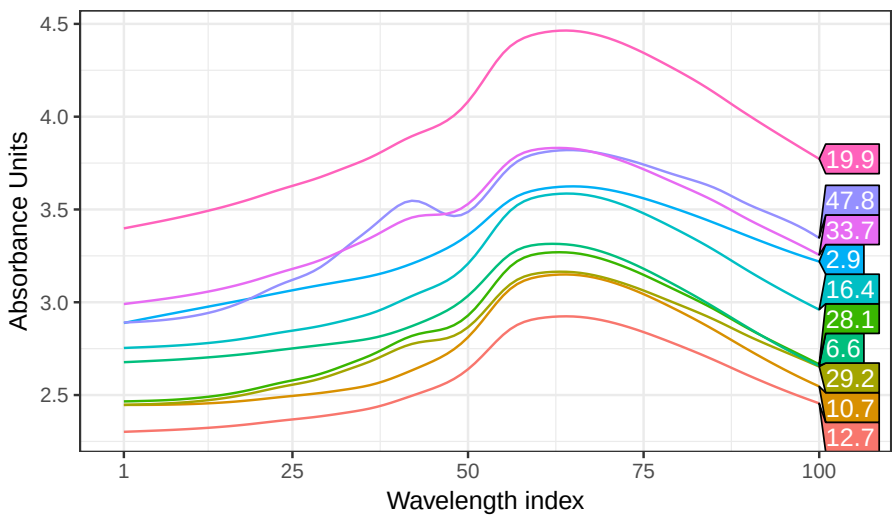


Figure 4.9: Sample of spectrometric curves used to predict fat content of meat. For each meat sample the data consists of a 100 channel spectrum of absorbances and the contents of moisture, fat (numbers shown in boxes) and protein measured in percent. The absorbance is $-\log 10$ of the transmittance measured by the spectrometer. The three contents, measured in percent, are determined by analytic chemistry.

fig:tecator.data

focus here is to use the **iprior** package to fit several I-prior models to the Tecator data set, and calculate out-of-sample predictive error rates. We compare the predictive performance of I-prior models against Gaussian process regression and the many other different methods applied on this data set. These methods include neural networks (Thodberg, 1996), kernel smoothing (Ferraty and Vieu, 2006), single and multiple index functional regression models (Chen et al., 2011), sliced inverse regression (SIR) and sliced average variance estimation (SAVE), multivariate adaptive regression splines (MARS), partial least squares (PLS), and functional additive model with and without component selection (FAM & CSEFAM). An analysis of this data set using the SIR and SAVE methods were conducted by Lian and Li (2014), while the MARS, PLS and (CSE)FAM methods were studied by Zhu et al. (2014). Table 4.5 tabulates the results of all of these methods from the various references.

Assuming a regression model as in (4.6), we would like to model the `fat` content y_i using the spectral curves x_i. Let $x_i(t)$ denote the absorbance for wavelength $t = 1, \ldots, 100$. From Figure 4.9, it appears that the curves are smooth enough to be differentiable, and therefore it is reasonable to assume that they lie in the Sobolev-Hilbert space as discussed in Section Section 4.1.5. We take first differences of the 100-dimensional matrix, which

leaves us with the 99-dimensional covariate saved in the object named `absorp`. The `fat` and `absorp` data have been split into `*.train` and `*.test` samples, as mentioned earlier. Our first modelling attempt is to fit a linear effect by regressing the responses `fat.train` against a single high-dimensional covariate `absorp.train` using the linear RKHS and the direct optimisation method.

```
R> # Model 1: Canonical RKHS (linear)
R> (mod1 <- iprior(y = fat.train, absorp.train))

## iter   10 value 222.653144
## final  value 222.642108
## converged
## Log-likelihood value: -445.2844
##
##     lambda        psi
## 4576.86595    0.11576
```

Our second and third model uses polynomial RKHSs of degrees two and three, which allows us to model quadratic and cubic terms of the spectral curves respectively. We also opted to estimate a suitable offset parameter, and this is called to `iprior()` with the option `est.offset = TRUE`. Each of the two models has a single scale parameter, an offset parameter, and an error precision to be estimated. The direct optimisation method has been used, and while both models converged regularly, it was noticed that there were multiple local optima that hindered the estimation (output omitted).

```
R> # Model 2: Polynomial RKHS (quadratic)
R> mod2 <- iprior(y = fat.train, absorp.train, kernel = "poly2",
+                 est.offset = TRUE)
R> # Model 3: Polynomial RKHS (cubic)
R> mod3 <- iprior(y = fat.train, absorp.train, kernel = "poly3",
+                 est.offset = TRUE)
```

Next, we attempt to fit a smooth dependence of fat content on the spectrometric curves using the fBm RKHS. By default, the Hurst coefficient for the fBm RKHS is set to be 0.5. However, with the option `est.hurst = TRUE`, the Hurst coefficient is included in the estimation procedure. We fit models with both a fixed value for Hurst (at 0.5) and an estimated value for Hurst. For both of these models, we encountered numerical issues when using the direct optimisation method. The L-BFGS algorithm kept

on pulling the hyperparameter towards extremely high values, which in turn made the log-likelihood value greater than the machine's largest normalised floating-point number (`.Machine$double.xmax = 1.797693e+308`). Investigating further, it seems that estimates at these large values give poor training and test error rates, though likelihood values here are high (local optima). To get around this issue, we used the EM algorithm to estimate the fixed Hurst model, and the `mixed` method for the estimated Hurst model. For both models, the `stop.crit` was relaxed and set to `1e-3` for quicker convergence, though this did not affect the predictive abilities compared to a more stringent `stop.crit`.

```
R> # Model 4: fBm RKHS (default Hurst = 0.5)
R> (mod4 <- iprior(y = fat.train, absorp.train, kernel = "fbm",
+                  method = "em", control = list(stop.crit = 1e-3)))
```

```
## ============================================
## Converged after 65 iterations.
## Log-likelihood value: -204.4592
##
##     lambda        psi
##    3.24112 1869.32897
```

```
R> # Model 5: fBm RKHS (estimate Hurst)
R> (mod5 <- iprior(fat.train, absorp.train, kernel = "fbm", method = "mixed",
+                  est.hurst = TRUE, control = list(stop.crit = 1e-3)))
```

```
## Running 5 initial EM iterations
## ================================================================
## Now switching to direct optimisation
## iter   10 value 115.648462
## final  value 115.645800
## converged
## Log-likelihood value: -231.2923
##
##    lambda     hurst       psi
## 204.97184   0.70382   9.96498
```

Finally, we fit an I-prior model using the SE RKHS with lengthscale estimated. Here we illustrate the use of the `restarts` option, in which the model is fitted repeatedly from different starting points. In this case, eight random initial parameter values were used and these jobs were parallelised across the eight available cores of the machine.

The additional `par.maxit` option in the `control` list is an option for the maximum number of iterations that each parallel job should do. We have set it to 100, which is the same number for `maxit`, but if `par.maxit` is less than `maxit`, the estimation procedure continues from the model with the best likelihood value. We see that starting from eight different initial values, direct optimisation leads to (at least) two log-likelihood optima sites, -231.5 and -680.5.

```
R> # Model 6: SE kernel
R> (mod6 <- iprior(fat.train, absorp.train, est.lengthscale = TRUE,
+                  kernel = "se", control = list(restarts = TRUE,
+                                                par.maxit = 100)))
## Performing 8 random restarts on 8 cores
## ========================================================================
## Log-likelihood from random starts:
##     Run 1     Run 2     Run 3     Run 4     Run 5     Run 6     Run 7
## -680.4637 -231.5440 -231.5440 -231.5440 -231.5440 -680.4637 -680.4637
##     Run 8
## -231.5440
## Continuing on Run 3
## final  value 115.771932
## converged
## Log-likelihood value: -231.544
##
##      lambda lengthscale         psi
##    96.10718     0.09269     6.15429
```

Predicted values of the test data set can be obtained using the `predict()` function. An example for obtaining the first model's predicted values is shown below. The `predict()` method for `ipriorMod` objects also return the test MSE if the vector of test data is supplied.

```
R> predict(mod1, newdata = list(absorp.test), y.test = fat.test)
## Test RMSE: 2.890353
##
## Predicted values:
##  [1] 43.607 20.444  7.821  4.491  9.044  8.564  7.935 11.615 13.807
## [10] 17.359
## # ... with 33 more values
```

Table 4.5: A summary of the root mean squared error (RMSE)of prediction for the I-prior models and various other methods in literature conducted on the Tecator data set. Values for the methods under *Others* were obtained from the corresponding references cited earlier.

tab:tecator

	RMSE	
Model	Train	Test
I-prior		
Linear	2.89	2.89
Quadratic	0.72	0.97
Cubic	0.37	0.58
Smooth (fBm-0.50)	0.00	0.68
Smooth (fBm-0.70)	0.19	0.63
Smooth (SE-0.09)	0.35	1.85
Gaussian process regression		
Linear	0.18	2.36
Smooth (SE-7.04)	0.17	2.10
Others		
Neural network[a]		0.36
Kernel smoothing[b]		1.49
Single/multiple indices model[c]		1.55
Sliced inverse regression		0.90
Sliced average variance estimation		1.70
MARS[d]		0.88
Partial least squares[d]		1.01
CSEFAM[d]		0.85

[a] Neural network best results with automatic relevance determination (ARD) quoted.
[b] Data set used was a 160/55 training/test split.
[c] These are results of a leave-one-out cross-validation scheme.
[d] Data set used was an extended version with $n = 240$, and a random 185/55 training/test split.

These results are summarised in Table 4.5. For the I-prior models, a linear effect of the functional covariate gives a training RMSE of 2.89, which is improved by both the qudratic and cubic model. The training RMSE is improved further by assuming a smooth RKHS of functions for f, i.e. the fBm and SE RKHSs. When it comes to out-of-sample test error rates, the cubic model gives the best RMSE out of the I-prior models for this particular data set, with an RMSE of 0.58. This is followed closely by the

fBm RKHS with estimated Hurst coefficient (fBm-0.70) and also the fBm RKHS with default Hurst coefficient (fBm-0.50). The best performing I-prior model is only outclassed by the neural networks of Thodberg (1996), who also performed model selection using automatic relevance determination (ARD). The I-prior models also give much better test RMSE than Gaussian process regression[5].

4.6 Conclusion

The steps for I-prior modelling are essentially three-fold:

1. Select an appropriate function space (equivalently, kernels) for which specific effects are desired on the covariates.
2. Estimate the posterior regression function and optimise the hyperparameters, which include the RKHS scale parameter(s), error precision, and any other kernel parameters such as the Hurst index.
3. Perform post-estimation procedures such as
    - Posterior predictive checks;
    - Model comparison via log-likelihood ratio tests/empirical Bayes factors; and
    - Prediction of new data point.

The main sticking point with the estimation procedure is the involvement of the $n \times n$ kernel matrix, for which its inverse is needed. This requires $O(n^2)$ storage and $O(n^3)$ computational time. The computational issue faced by I-priors are mirrored in Gaussian process regression, so the methods to overcome these computational challenges in GPR can be explored further. However, most efficient computational solutions exploit the nature of the SE kernel structure, which is the most common kernel used in GPR. Nonetheless, we suggest the following as considerations for future work:

1. **Sparse variational approximations**. Variational methods have seen an active development in recent times. By using inducing points (Titsias, 2009) or stochastic variational inference (Hensman et al., 2013), such methods can greatly reduce computational storage and speed requirements. A recent paper by Cheng and Boots

[5] GPR models were fit using `gausspr()` in **kernlab**.

(2017) also suggests a variational algorithm with linear complexity for GPR-type models.

2. **Accelerating the EM algorithm further**. Two methods can be explored. The first is called parameter-expansion EM algorithm (PXEM) by (Liu et al., 1998), which has been shown to be promising for random-effects type models. It involves correcting the M-step by a 'covariance adjustment', so that extra information can be capitalised on to improve convergence rates. The second is a quasi-Newton acceleration of the EM algorithm as proposed by Lange (1995). A slight change to the EM gradient algorithm in the M-step steers the EM algorithm to the Newton-Raphson algorithm, thus exploiting the benefits of the EM algorithm in the early stages (monotonic increase in likelihood) and avoiding the pitfalls of Newton-Raphson (getting stuck in local optima). Both algorithms require an in-depth reassessment of the EM algorithm to be tailored to I-prior models.

4.7 Miscellanea

4.7.1 Similarity to the g-prior

misc:gprior

The I-prior for $\boldsymbol{\beta}$ resembles the objective g-prior (Zellner, 1986) for regression coefficients,

$$\boldsymbol{\beta} \sim \mathrm{N}_p\left(\mathbf{0}, g(\mathbf{X}^\top \boldsymbol{\Psi} \mathbf{X})^{-1}\right),$$

although they are quite different objects. The g-prior for $\boldsymbol{\beta}$ has the *inverse* (scaled) Fisher information matrix as its covariance matrix. This, in itself, has a much different and arguably counterintuitive meaning: large amounts of Fisher information about $\boldsymbol{\beta}$ corresponds to a small prior variance, and hence less deviation away from the prior mean of zero in estimating $\boldsymbol{\beta}$. The choice of the hyperparameter g has been the subject of much debate, with choices ranging from fixing $g = n$ (corresponding to the concept of *unit Fisher information*), to fully Bayesian and empirical Bayesian methods of estimating g from the data.

On the other hand, we note that the g-prior has an I-prior interpretation when argues as follows. Assume that the regression function f lies in the continual dual space of $\mathbb{R}^p$ equipped with the inner product $\langle \mathbf{x}, \mathbf{x}' \rangle_{\mathcal{X}} = \mathbf{x}^\top (\mathbf{X}^\top \boldsymbol{\Psi} \mathbf{X})^{-1} \mathbf{x}$. With this inner product

and from (3.3) (p. 12), the Fisher information for $\boldsymbol{\beta}$ is

$$\begin{aligned}\mathcal{I}_g(\boldsymbol{\beta}) &= \sum_{i=1}^{n}\sum_{j=1}^{n} \psi_{ij}(\mathbf{X}^\top\boldsymbol{\Psi}\mathbf{X})^{-1}\mathbf{x}_i \otimes (\mathbf{X}^\top\boldsymbol{\Psi}\mathbf{X})^{-1}\mathbf{x}_j \\ &= (\mathbf{X}^\top\boldsymbol{\Psi}\mathbf{X})^{-1}\cancel{(\mathbf{X}^\top\boldsymbol{\Psi}\mathbf{X})\,(\mathbf{X}^\top\boldsymbol{\Psi}\mathbf{X})^{-1}} \\ &= (\mathbf{X}^\top\boldsymbol{\Psi}\mathbf{X})^{-1},\end{aligned}$$

and this, rather than the usual $\mathbf{X}^\top\boldsymbol{\Psi}\mathbf{X}$ as the prior covariance matrix for $\boldsymbol{\beta}$, means that the I-prior is in fact the standard g-prior.

The metric induced by the inner product is actually the *Mahalanobis distance*, a scale-invariant natural distance if the covariates are measured on different scales. To expand on this idea, circle back to the regression function and write it as $f(\mathbf{x}) = \langle \mathbf{x}, \boldsymbol{\beta}\rangle_{\mathcal{X}}$. In usual least squares regression, the choice of inner product is irrelevant, so the usual dot product is commonly used (however, as we have seen above, the choice of inner product determines the form of the Fisher information for $\boldsymbol{\beta}$). In particular, suppose that all the x_{ik}'s, $k = 1, \ldots, p$ for each unit $i = 1, \ldots, n$ are measured on the same scale; for instance, these could be measurements in centimetres. In this case, the dot product is reasonable, because $\langle \mathbf{x}_i, \mathbf{x}_j\rangle = \sum_{k=1}^{p} x_{ik}x_{jk}$ and the inner product has a coherent unit, namely the squared unit of the x_{ik}'s. However, if they were a mix of various scaled measurements, then obviously the inner product's unit is incoherent—one would be resorted to adding measurements in different units, for example, cm^2 and kg^2 and so on. In such a case, a unitless inner product is appropriate, like the Mahalonobis inner product, which technically rescales the x_{ik}'s to unity. In summary, if the covariates are all measured on the same scale, then the I-prior is appropriate, and if not, the g-prior is appropriate.

misc:multilevelmodels

4.7.2 Multilevel models

Write $\alpha = \beta_0$, and for simplicity, assume iid errors, i.e., $\boldsymbol{\Psi} = \psi\mathbf{I}_n$. The form of $f \in \mathcal{F}$ is now $f(\mathbf{x}_i^{(j)}, j) = \sum_{i'=1}^{n_{j'}}\sum_{j'=1}^{m} h_\lambda\big((\mathbf{x}_i^{(j)}, j), (\mathbf{x}_{i'}^{(j')}, j')\big)w_{i'j'}$, where each $w_{i'j'} \sim \text{N}(0, \psi^{-1})$.

Now, functions in the scaled RKHS $\mathcal{F}_2$ have the form

$$\begin{aligned} f_2(j) &= \sum_{i=1}^{n_{j'}} \sum_{j'=1}^{m} \lambda_2 \left(\frac{\delta_{jj'}}{p_j} - 1 \right) w_{ij'} \\ &= \lambda_2 \left(\frac{w_{+j}}{p_j} - w_{++} \right), \end{aligned}$$

where a '+' in the index of w_{ik} indicates a summation over that index, and p_j is the empirical distribution over $\mathcal{M}$, i.e. $p_j = n_j/n$. Clearly $f_2(j)$ is a variable depending on j, so write $f_2(j) = \beta_{0j}$. The distribution of β_{0j} is normal with zero mean and variance

$$\begin{aligned} \operatorname{Var} \beta_{0j} &= \lambda_2^2 \left(\frac{\cancel{n_j}\psi}{\cancel{n_j^2}/n^2} + n\psi \right) \\ &= n\psi\lambda_2^2 \left(\frac{1}{p_j} + 1 \right). \end{aligned}$$

The covariance between any two random intercepts β_{0j} and $\beta_{0j'}$ is

$$\begin{aligned} \operatorname{Cov}(\beta_{0j}, \beta_{0j'}) &= \operatorname{Cov}\left(\lambda_2 \left(\frac{w_{+j}}{p_j} - w_{++} \right), \lambda_2 \left(\frac{w_{+j'}}{p_{j'}} - w_{++} \right) \right) \\ &= \frac{\lambda_2^2}{p_j p_{j'}} \cancelto{0}{\operatorname{Cov}(w_{+j}, w_{+j'})} - \frac{\lambda_2^2}{p_j} \operatorname{Cov}(w_{+j}, w_{++}) - \frac{\lambda_2^2}{p_{j'}} \operatorname{Cov}(w_{++}, w_{+j'}) \\ &\quad + \lambda_2^2 \operatorname{Cov}(w_{++}, w_{++}) \\ &= -\frac{\lambda_2^2}{\cancel{n_j}/n} \cancel{n_j}\psi - \frac{\lambda_2^2}{\cancel{n_{j'}}/n} \cancel{n_{j'}}\psi + \lambda_2^2 n\psi \\ &= -n\psi\lambda_2^2. \end{aligned}$$

Functions in $\mathcal{F}_{12}$, on the other hand, have the form

$$\begin{aligned} f_{12}(\mathbf{x}_i, j) &= \sum_{i'=1}^{n_{j'}} \sum_{j'=1}^{m} \lambda_1\lambda_2 \cdot \tilde{\mathbf{x}}_i^{(j)\top} \tilde{\mathbf{x}}_{i'}^{(j')} \cdot \left(\frac{\delta_{jj'}}{p_j} - 1 \right) w_{i'j'} \\ &= \tilde{\mathbf{x}}_i^{(j)\top} \underbrace{\left(\frac{\lambda_1\lambda_2}{p_j} \sum_{i'=1}^{n_j} \tilde{\mathbf{x}}_{i'}^{(j)} w_{i'j} - \lambda_1\lambda_2 \sum_{i'=1}^{n_{j'}} \sum_{j'=1}^{m} \tilde{\mathbf{x}}_{i'}^{(j')} w_{i'j'} \right)}_{\boldsymbol{\beta}_{1j}}, \end{aligned}$$

and this is, as expected, a linear form dependent on cluster j. We can calculate the

variance for β_{1j} to be

$$\begin{aligned}
\operatorname{Var} \boldsymbol{\beta}_{1j} &= \lambda_1^2 \lambda_2^2 \operatorname{Var} \left(\frac{1}{p_j} \tilde{\mathbf{X}}_j^\top \mathbf{w}_j - \tilde{\mathbf{X}}^\top \mathbf{w} \right) \\
&= \lambda_1^2 \lambda_2^2 \left(\frac{\psi}{n_j^2/n^2} \tilde{\mathbf{X}}_j^\top \tilde{\mathbf{X}}_j + \psi \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} - \frac{1}{p_j} \tilde{\mathbf{X}}_j^\top \operatorname{Cov}(\mathbf{w}_j, \mathbf{w}) \tilde{\mathbf{X}}^\top \right) \\
&= n\psi \lambda_1^2 \lambda_2^2 \left(\frac{1}{p_j} \mathbf{S}_j + \mathbf{S} - \mathbf{S}_j \right) \\
&= n\psi \lambda_1^2 \lambda_2^2 \left(\left(\frac{1}{p_j} - 1 \right) \mathbf{S}_j + \mathbf{S} \right)
\end{aligned}$$

where $\mathbf{S}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} (\mathbf{x}_i^{(j)} - \bar{\mathbf{x}})^\top (\mathbf{x}_i^{(j)} - \bar{\mathbf{x}})$, $\mathbf{S} = \frac{1}{n} \sum_{i=1}^{n_j} \sum_{j=1}^{m} (\mathbf{x}_i^{(j)} - \bar{\mathbf{x}})^\top (\mathbf{x}_i^{(j)} - \bar{\mathbf{x}})$, and $\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n_j} \sum_{j=1}^{m} \mathbf{x}_i^{(j)}$. The covariance between two vectors of the random slopes is

$$\begin{aligned}
\operatorname{Cov}(\boldsymbol{\beta}_{1j}, \boldsymbol{\beta}_{1j'}) &= \lambda_1^2 \lambda_2^2 \operatorname{Cov} \left(\frac{1}{p_j} \tilde{\mathbf{X}}_j^\top \mathbf{w}_j - \tilde{\mathbf{X}}^\top \mathbf{w}, \frac{1}{p_{j'}} \tilde{\mathbf{X}}_{j'}^\top \mathbf{w}_{j'} - \tilde{\mathbf{X}}^\top \mathbf{w} \right) \\
&= \psi \lambda_1^2 \lambda_2^2 \left(\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} - \frac{1}{p_j} \tilde{\mathbf{X}}_j^\top \tilde{\mathbf{X}}_j - \frac{1}{p_{j'}} \tilde{\mathbf{X}}_{j'}^\top \tilde{\mathbf{X}}_{j'} \right) \\
&= n\psi \lambda_1^2 \lambda_2^2 \left(\mathbf{S} - \mathbf{S}_j - \mathbf{S}_{j'} \right).
\end{aligned}$$

Another quantity of interest is the covariance between the random intercepts and random slopes:

$$\begin{aligned}
\operatorname{Cov}(\beta_{0j}, \boldsymbol{\beta}_{1j}) &= \lambda_1 \lambda_2^2 \operatorname{Cov} \left(\frac{1}{p_j} \mathbf{1}_{n_j}^\top \mathbf{w}_j - \mathbf{1}_n^\top \mathbf{w}, \frac{1}{p_j} \tilde{\mathbf{X}}_j^\top \mathbf{w}_j - \tilde{\mathbf{X}}^\top \mathbf{w} \right) \\
&= \psi \lambda_1 \lambda_2^2 \left(\cancel{\mathbf{1}_n^\top \tilde{\mathbf{X}}}^{0} + \frac{1}{p_j^2} \mathbf{1}_{n_j}^\top \tilde{\mathbf{X}}_j - \frac{2}{p_j} \mathbf{1}_{n_j}^\top \tilde{\mathbf{X}}_j \right) \\
&= n\psi \lambda_1 \lambda_2^2 \left(\left(\frac{1}{p_j} - 2 \right) \frac{1}{n_j} \sum_{i=1}^{n_j} (\mathbf{x}_i^{(j)} - \bar{\mathbf{x}}) \right) \\
&= n\psi \lambda_1 \lambda_2^2 \left(\frac{1}{p_j} - 2 \right) (\bar{\mathbf{x}}^{(j)} - \bar{\mathbf{x}})
\end{aligned}$$

and

$$\begin{aligned}
\operatorname{Cov}(\beta_{0j}, \boldsymbol{\beta}_{1j'}) &= \lambda_1\lambda_2^2 \operatorname{Cov}\left(\frac{1}{p_j}\mathbf{1}_{n_j}^\top \mathbf{w}_j - \mathbf{1}_n^\top \mathbf{w}, \frac{1}{p_{j'}}\tilde{\mathbf{X}}_{j'}^\top \mathbf{w}_{j'} - \tilde{\mathbf{X}}^\top \mathbf{w}\right) \\
&= \psi\lambda_1\lambda_2^2 \left(\cancelto{0}{\mathbf{1}_n^\top \tilde{\mathbf{X}}} + \frac{1}{p_j p_{j'}}\mathbf{1}_{n_j}^\top \cancelto{0}{\operatorname{Cov}(\mathbf{w}_j, \mathbf{w}_{j'})} \tilde{\mathbf{X}}_{j'} - \frac{1}{p_j}\mathbf{1}_{n_j}^\top \tilde{\mathbf{X}}_j - \frac{1}{p_{j'}}\mathbf{1}_{n_{j'}}^\top \tilde{\mathbf{X}}_{j'}\right) \\
&= n\psi\lambda_1\lambda_2^2 \left(-\frac{1}{n_j}\sum_{i=1}^{n_j}(\mathbf{x}_i^{(j)} - \bar{\mathbf{x}}) - \frac{1}{n_{j'}}\sum_{i=1}^{n_{j'}}(\mathbf{x}_i^{(j')} - \bar{\mathbf{x}})\right) \\
&= n\psi\lambda_1\lambda_2^2 \left(2\bar{\mathbf{x}} - \bar{\mathbf{x}}^{(j)} - \bar{\mathbf{x}}^{(j')}\right).
\end{aligned}$$

4.7.3 A brief introduction to Hamiltonian Monte Carlo

misc:hmc

Hamiltonian Monte Carlo had its beginnings in statistical physics, with the 1987 paper by Duane et al. using what they called 'Hybrid Monte Carlo' in lattice models of quantum theory. Their work merged the approaches of molecular dynamics and Markov chain Monte Carlo methods. As interesting side note, their method abbreviates also to 'HMC', but throughout the statistical literature, it is more commonly referred to by its more descriptive name Hamiltonian Monte Carlo. Incidentally, the use of HMC started with applications to neural networks as early as 1996 (see Neal et al. (2011) for an excellent review of the subject matter). It was not until 2011 when active development of the method, and in particular, software for for statistical applications began. The Stan initiative (Carpenter et al., 2017) began in response to difficulties faced when performing full Bayesian inference on multilevel generalised linear models. These difficulties mainly involved poor efficiency in usual MCMC samplers, particularly high autocorrelations in the posterior chains, which meant that many chains and many iterations were required to get an adequate sample. It was a case of exhausting all possible algorithmic remedies for existing samplers (Gibbs samplers, Metropolis samplers, etc.), and realising that fundamentally not much improvement can be had unless a novel sampling technique was discovered.

The basic idea behind HMC is to use Hamiltonian dynamics to propose new states in the posterior sampling, rather than relying on 'random walks'. If one were to understand and use the geometry of the posterior density to one's benefit, then it should be possible to generate new proposal states with high probabilities of acceptance and move far away from the current state. Hamiltonian dynamics, like classical Newtonian mechanics, provides a framework for modelling the motion of a body in space across

time t. Additionally, Hamiltonian dynamics concatenates the position vector x with its momentum z, and the motion of x in d-dimensional space is then described through Hamilton's equations

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\partial H}{\partial z} \quad \text{and} \quad \frac{\mathrm{d}z}{\mathrm{d}t} = -\frac{\partial H}{\partial x}, \tag{4.18}$$

{eq:hamilton1}

where $H = H(x, z)$ is called the Hamiltonian of the system. The Hamiltonian is an operator which encapsulates the total energy of the system. In a closed system, one can express the sum of operators corresponding to the kinetic energy $K(p)$ and the potential energy $U(z)$ of the system

$$H(x, z) = K(z) + U(x). \tag{4.19}$$

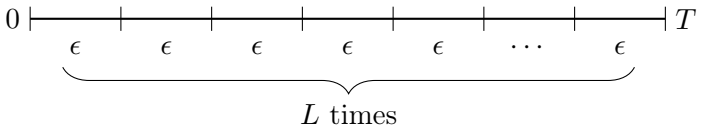
{eq:hamilton2}

Substituing (4.19) into (4.18), we get the system of partial differential equations (PDEs)

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\partial}{\partial z}K(z) \quad \text{and} \quad \frac{\mathrm{d}z}{\mathrm{d}t} = -\frac{\partial}{\partial x}U(x). \tag{4.20}$$

{eq:hamilton3}

To describe the evolution of $\big(x(t), z(t)\big)$ from time t to $t+T$, it is necessary to discretise time, and split $T = L\epsilon$. The quantity L is known as the number of *leapfrogs*, and ϵ the *step size*.



The system of PDEs is solved using Euler's method, or the more commonly used leapfrog integration, which is a three-step process:

1. **Half-step momentum.** $z(t + \epsilon/2) = z(t) - \frac{\epsilon}{2}\frac{\partial}{\partial x}U\big(x(t)\big)$
2. **Full-step position.** $x(t + \epsilon) = x(t) + \epsilon\frac{\partial}{\partial z}K\big(z(t + \epsilon/2)\big)$
3. **Half-step momentum.** $z(t + \epsilon) = z(t + \epsilon/2) = z(t) - \frac{\epsilon}{2}\frac{\partial}{\partial x}U\big(x(t)\big)$

in which steps 1–3 are repeated L times.

Having knowing the formula for how particles move in space, we can use this information to treat random points drawn from some probability density as 'particles'. Randomness of position and momentum are prescribed through probability densities on each. Given some energy function $E(\theta)$ over states θ, the *canonical distribution* of the states θ (otherwise known as the *canonical ensemble*) is given by the probability density

function

$$p(\theta) \propto \exp\left(-\frac{E(\theta)}{k\tau}\right),$$

where k is Boltzmann's constant, τ is the absolute temperature of the system. The Hamiltonian is one such energy function over states (x, z). By replacing $E(\theta)$ by (4.19) in the pdf above, we realise that the distribution for x and z are independent. The system can be manipulated such that $k\tau = 1$—in any case, these are constants which can be absorbed into one of the terms in the pdf anyway.

Using a *quadratic kinetic energy* function $K(z) = z^\top M^{-1} z/2$[6], we find that the probability density function for z is

$$p(z) \propto \exp\left(-\frac{1}{2} z^\top M^{-1} z\right),$$

implying $z \sim \mathrm{N}_d(0, M)$. Here, $M = \mathrm{diag}(m_1, \ldots, m_d)$ is called the *mass matrix*, which obviously serves as the variance for the randomly distributed z. As for the potential energy, choose a function such that $U(x) = -\log p(x)$, implying $p(x) \propto \exp\big(-U(x)\big)$. Here, $p(x)$ represents the target density from which we wish to sample, for instance, a posterior density of interest. Thus, to sample variables x from $p(x)$, one artificially introduces momentum variables z and sample jointly instead from $p(x, z) = p(z)p(z)$, and discarding z thereafter. The HMC algorithm is summarised in Algorithm 2.

Algorithm 2 Hamiltonian Monte Carlo

alg:hmc

1: **initialise** $x^{(0)}$, $z^{(0)}$ and choose values for L, ϵ and M
2: **while** not converged **do**
3: Draw $z \sim \mathrm{N}_d(0, M)$ ▷ Perturb momentum
4: Move $(x^{(t)}, z^{(t)}) \mapsto (x^*, z^*)$ using Hamiltonian dynamics ▷ Proposal state
5: Accept/reject proposal state, i.e. ▷ Metropolis update

$$(x^{(t+1)}, z^{(t+1)}) \leftarrow \begin{cases} (x^*, z^*) & \text{w.p. } \min(1, A) \\ (x^{(t)}, z^{(t)}) & \text{otherwise} \end{cases}$$

where

$$A = \frac{p(x^*, z^*)}{(x^{(t)}, z^{(t)})} = \exp\Big(H(x, z) - H(x^{(t)}, z^{(t)})\Big)$$

6: **end while**
7: **return** Samples $\big\{x^{(t)} \,|\, t = 1, 2, \ldots\big\}$

HMC is often times superior to standard Gibbs sampling, for a variety of reasons.

For one, conjugacy does not play any role in the efficiency of the HMC sampler, thus freeing the modeller to choose more appropriate and more intuitive prior densities for the parameters of the model. For another, the HMC sampler is designed to incite little autocorrelations between samples, and thus increasing efficiency.

Several drawbacks do exist with the HMC sampler. Firstly, it is impossible to directly sample from discrete distributions $p(x)$. More concretely, HMC requires that the domain of $p(x)$ is continuous and that $\partial \log p(x)/\partial x$ is inexpensive to compute. To work around this, one must reformulate the model by marginalising out the discrete variables, and obtain them back later by separately sampling from their posteriors. Alternatively, a Gibbs sampler specifically for the discrete variables could be augmented with the HMC sampler. The other drawback of HMC is that there are many tuning parameters (leapfrog L, step-size ϵ, mass matrix M, etc.) that is not immediately easy to perfect, at least not to the novice user.

The implementation of HMC by the programming language Stan, which interfaces many other programming languages including R, Python, MATLAB, Julia, Stata and Mathematica, is a huge step forward in computational Bayesian analysis. Stan takes the liberty of performing all the tuning necessary, and the practitioner is left with simply specifying the model. A vast library of differentiable probability functions are available, with the ability to bring your own code as well. Development is very active and many improvements and optimisations have been made since its inception.

Appendix

4.8 Deriving the posterior distribution for w

apx:posterior

In the following derivation, we implicitly assume the dependence on $\mathbf{f}_0$ and θ. The distribution of $\mathbf{y}|\mathbf{w}$ is $\mathrm{N}_n(\boldsymbol{\alpha}+\mathbf{f}_0+\mathbf{H}_\eta\mathbf{w}, \boldsymbol{\Psi}^{-1})$, where $\boldsymbol{\alpha} = \alpha\mathbf{1}_n$, while the prior distribution

[6] Thinking back to elementary mechanics, this is the familiar $\frac{1}{2}mv^2$ formula for kinetic energy and substituting in the identity $z = mv$, where m is the mass of the object, and v is its velocity.

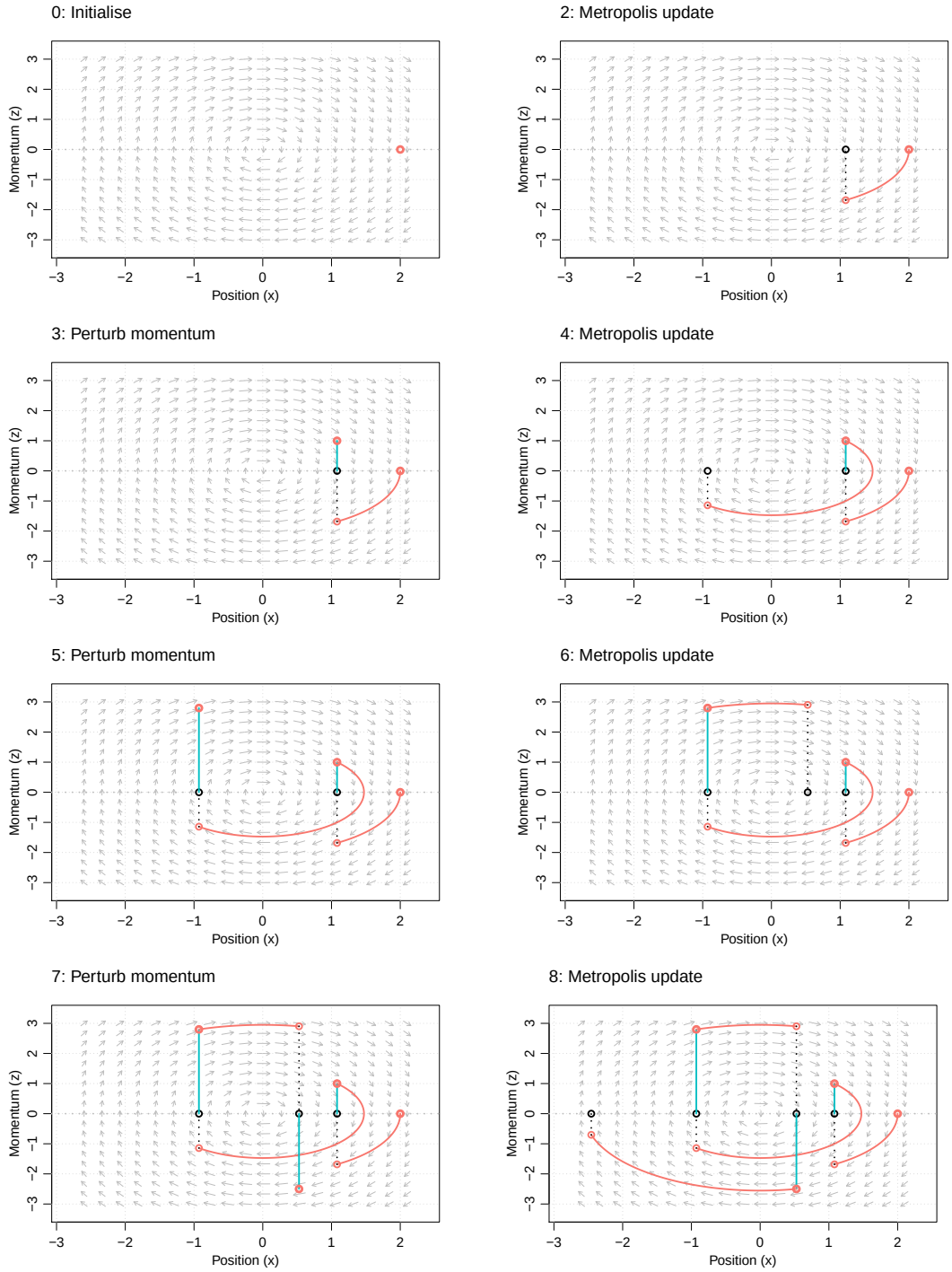


Figure 4.10: A phase diagram schematic of HMC sampling in one-dimension. At step 0, initialise values for momentum and position. At step 1, simulate movement using Hamiltonian dynamics, accept position and discard momentum. At step 2, perturb momentum using a normal density, then repeat.

for $\mathbf{w}$ is $\mathrm{N}_n(\mathbf{0}, \boldsymbol{\Psi})$. Since $p(\mathbf{w}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{w})p(\mathbf{w})$, we have that

$$\begin{aligned}
\log p(\mathbf{w}|\mathbf{y}) &= \log p(\mathbf{y}|\mathbf{w}) + \log p(\mathbf{w}) \\
&= \text{const.} + \cancel{\frac{1}{2}\log|\boldsymbol{\Psi}|} - \frac{1}{2}(\mathbf{y} - \boldsymbol{\alpha} - \mathbf{f}_0 - \mathbf{H}_\eta\mathbf{w})^\top\boldsymbol{\Psi}(\mathbf{y} - \boldsymbol{\alpha} - \mathbf{f}_0 - \mathbf{H}_\eta\mathbf{w}) \\
&\quad - \cancel{\frac{1}{2}\log|\boldsymbol{\Psi}|} - \frac{1}{2}\mathbf{w}^\top\boldsymbol{\Psi}^{-1}\mathbf{w} \\
&= \text{const.} - \frac{1}{2}\mathbf{w}^\top(\mathbf{H}_\eta\boldsymbol{\Psi}\mathbf{H}_\eta + \boldsymbol{\Psi}^{-1})\mathbf{w} + (\mathbf{y} - \boldsymbol{\alpha} - \mathbf{f}_0)^\top\boldsymbol{\Psi}\mathbf{H}_\eta\mathbf{w}.
\end{aligned}$$

Setting $\mathbf{A} = \mathbf{H}_\eta\boldsymbol{\Psi}\mathbf{H}_\eta + \boldsymbol{\Psi}^{-1}$, $\mathbf{a}^\top = (\mathbf{y} - \boldsymbol{\alpha} - \mathbf{f}_0)^\top\boldsymbol{\Psi}\mathbf{H}_\eta$, and using the fact that

$$\mathbf{w}^\top\mathbf{A}\mathbf{w} - 2\mathbf{a}^\top\mathbf{w} = (\mathbf{w} - \mathbf{A}^{-1}\mathbf{a})^\top\mathbf{A}(\mathbf{w} - \mathbf{A}^{-1}\mathbf{a}),$$

we have that $\mathbf{w}|\mathbf{y}$ is normally distributed with the required mean and variance.

Alternatively, one could have shown this using standard results of multivariate normal distributions. Noting that the covariance between $\mathbf{y}$ and $\mathbf{w}$ is

$$\begin{aligned}
\mathrm{Cov}(\mathbf{y}, \mathbf{w}) &= \mathrm{Cov}(\boldsymbol{\alpha} + \mathbf{f}_0 + \mathbf{H}_\eta\mathbf{w} + \boldsymbol{\epsilon}, \mathbf{w}) \\
&= \mathbf{H}_\eta\,\mathrm{Cov}(\mathbf{w}, \mathbf{w}) \\
&= \mathbf{H}_\eta\boldsymbol{\Psi}
\end{aligned}$$

and that $\mathrm{Cov}(\mathbf{w}, \mathbf{y}) = \boldsymbol{\Psi}\mathbf{H}_\eta = \mathbf{H}_\eta\boldsymbol{\Psi} = \mathrm{Cov}(\mathbf{y}, \mathbf{w})$ by symmetry, the joint distribution $(\mathbf{y}, \mathbf{w})$ is

$$\begin{pmatrix}\mathbf{y}\\ \mathbf{w}\end{pmatrix} \sim \mathrm{N}_{n+n}\left(\begin{pmatrix}\boldsymbol{\alpha} + \mathbf{f}_0\\ \mathbf{0}\end{pmatrix}, \begin{pmatrix}\mathbf{V}_y & \mathbf{H}_\eta\boldsymbol{\Psi}\\ \mathbf{H}_\eta\boldsymbol{\Psi} & \boldsymbol{\Psi}\end{pmatrix}\right).$$

Thus,

$$\begin{aligned}
\mathrm{E}[\mathbf{w}|\mathbf{y}] &= \mathrm{E}\,\mathbf{w} + \mathrm{Cov}(\mathbf{w}, \mathbf{y})(\mathrm{Var}\,\mathbf{y})^{-1}(\mathbf{y} - \mathrm{E}\,\mathbf{y}) \\
&= \mathbf{H}_\eta\boldsymbol{\Psi}\mathbf{V}_y^{-1}(\mathbf{y} - \boldsymbol{\alpha} - \mathbf{f}_0),
\end{aligned}$$

and

$$\begin{aligned}\operatorname{Var}[\mathbf{w}|\mathbf{y}] &= \operatorname{Var}\mathbf{w} - \operatorname{Cov}(\mathbf{w},\mathbf{y})(\operatorname{Var}\mathbf{y})^{-1}\operatorname{Cov}(\mathbf{y},\mathbf{w}) \\ &= \boldsymbol{\Psi} - \mathbf{H}_\eta\boldsymbol{\Psi}\mathbf{V}_y^{-1}\mathbf{H}_\eta\boldsymbol{\Psi} \\ &= \boldsymbol{\Psi} - \boldsymbol{\Psi}\mathbf{H}_\eta\left(\boldsymbol{\Psi}^{-1} + \mathbf{H}_\eta\boldsymbol{\Psi}\mathbf{H}_\eta\right)^{-1}\mathbf{H}_\eta\boldsymbol{\Psi} \\ &= \left(\boldsymbol{\Psi}^{-1} + \mathbf{H}_\eta\boldsymbol{\Psi}\mathbf{H}_\eta\right)^{-1} \\ &= \mathbf{V}_y^{-1}\end{aligned}$$

as a direct consequence of the Woodbury matrix identity.

4.9 A recap on the exponential family EM algorithm

apx:expem

Consider the density function $p(\cdot|\boldsymbol{\theta})$ of the complete data $\mathbf{z} = \{\mathbf{y}, \mathbf{w}\}$, which depends on parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_s)^\top \in \Theta \subseteq \mathbb{R}^s$, belonging to an exponential family of distributions. This density takes the form $p(\mathbf{z}|\boldsymbol{\theta}) = B(\mathbf{z})\exp\big(\langle\boldsymbol{\eta}(\boldsymbol{\theta}), \mathbf{T}(\mathbf{z})\rangle - A(\boldsymbol{\theta})\big)$, where $\boldsymbol{\eta} : \mathbb{R}^s \mapsto \mathbb{R}$ is a link function, $\mathbf{T}(\mathbf{z}) = \big(T_1(\mathbf{z}), \ldots, T_s(\mathbf{z})\big)^\top \in \mathbb{R}^s$ are the sufficient statistics of the distribution, and $\langle\cdot,\cdot\rangle$ is the usual Euclidean dot product. It is often easier to work in the *natural parameterisation* of the exponential family distribution

$$p(\mathbf{z}|\boldsymbol{\eta}) = B(\mathbf{z})\exp\big(\langle\boldsymbol{\eta}, \mathbf{T}(\mathbf{z})\rangle - A^*(\boldsymbol{\eta})\big) \tag{4.21}$$

{eq:pdfexpfamnat}

by defining $\boldsymbol{\eta} := \big(\eta_1(\boldsymbol{\theta}), \ldots, \eta_r(\boldsymbol{\theta})\big) \in \mathcal{E}$, and $\exp A^*(\boldsymbol{\eta}) = \int B(\mathbf{z})\exp\langle\boldsymbol{\eta}, \mathbf{T}(\mathbf{z})\rangle\,\mathrm{d}\mathbf{z}$ to ensure the density function normalises to one. As an aside, the set $\mathcal{E} := \big\{\boldsymbol{\eta} = (\eta_1, \ldots, \eta_s) \,|\, \int \exp A^*(\boldsymbol{\eta}) < \infty\big\}$ is called the *natural parameter space*. If $\dim\mathcal{E} = r < s = \dim\Theta$, then the the pdf belongs to the *curved exponential family* of distributions. If $\dim\mathcal{E} = r = s = \dim\Theta$, then the family is a *full exponential family*.

Assuming the latent $\mathbf{w}$ variables are observed and working with the natural parameterisation, then the complete maximum likelihood (ML) estimate for $\boldsymbol{\eta}$ is obtained by solving

$$\frac{\partial}{\partial\boldsymbol{\eta}}\log p(\mathbf{z}|\boldsymbol{\eta}) = \mathbf{T}(\mathbf{z}) - \frac{\partial}{\partial\boldsymbol{\eta}}A^*(\boldsymbol{\eta}) = 0. \tag{4.22}$$

{eq:expEM1}

Of course, the variable $\mathbf{w}$ are never observed, so the ML estimate for $\boldsymbol{\eta}$ can only be informed from what is observed. Let $p(\mathbf{y}|\boldsymbol{\eta}) = \int p(\mathbf{y}, \mathbf{w}|\boldsymbol{\eta})\,\mathrm{d}\mathbf{w}$ represent the marginal

density of the observations $\mathbf{y}$. Now, the ML estimate for $\boldsymbol{\eta}$ is obtained by solving

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\eta}} \log p(\mathbf{y}|\boldsymbol{\eta}) &= \frac{1}{p(\mathbf{y}|\boldsymbol{\eta})} \cdot \frac{\partial}{\partial \boldsymbol{\eta}} p(\mathbf{y}|\boldsymbol{\eta}) \\
&= \frac{1}{p(\mathbf{y}|\boldsymbol{\eta})} \cdot \frac{\partial}{\partial \boldsymbol{\eta}} \left(\int p(\mathbf{y}, \mathbf{w}|\boldsymbol{\eta}) \, \mathrm{d}\mathbf{w} \right) \\
&= \frac{1}{p(\mathbf{y}|\boldsymbol{\eta})} \cdot \int \left(\frac{\partial}{\partial \boldsymbol{\eta}} p(\mathbf{y}, \mathbf{w}|\boldsymbol{\eta}) \right) \mathrm{d}\mathbf{w} \\
&= \frac{1}{p(\mathbf{y}|\boldsymbol{\eta})} \cdot \int \left(p(\mathbf{y}, \mathbf{w}|\boldsymbol{\eta}) \frac{\partial}{\partial \boldsymbol{\eta}} \log p(\mathbf{y}, \mathbf{w}|\boldsymbol{\eta}) \right) \mathrm{d}\mathbf{w} \\
&= \int \left(\mathbf{T}(\mathbf{y}, \mathbf{w}) - \frac{\partial}{\partial \boldsymbol{\eta}} A^*(\boldsymbol{\eta}) \right) p(\mathbf{w}|\mathbf{y}, \boldsymbol{\eta}) \, \mathrm{d}\mathbf{w} \\
&= \mathrm{E}_{\mathbf{w}} \left[\mathbf{T}(\mathbf{y}, \mathbf{w}) | \mathbf{y} \right] - \frac{\partial}{\partial \boldsymbol{\eta}} A^*(\boldsymbol{\eta}) \qquad (4.23)
\end{aligned}$$

{eq:expEM2}

equated to zero. Note that we are allowed to change the order of integration and differentation provided the integrand is continuously differentiable. So the only difference between the first order condition of (4.22) and that of (4.23) is that the sufficient statistics involving the unknown $\mathbf{w}$ are replaced by their conditional or posterior expectations.

A useful identity to know is that $\frac{\partial}{\partial \boldsymbol{\eta}} A^*(\boldsymbol{\eta}) = \mathrm{E}_{\mathbf{z}} \, \mathbf{T}(\mathbf{z})$ (Casella and R. L. Berger, 2002, Theorem 3.4.2 & Exercise 3.32(a)), which can be expressed in terms of the original parameters $\boldsymbol{\theta}$. As a consequence, solving for the ML estimate for $\boldsymbol{\theta}$ from the FOC equations (4.23) is possible without having to deal with the derivative of A^* with respect to the natural parameters. Having said this, an analytical solution in $\boldsymbol{\theta}$ may not exist, because the relationship of $\boldsymbol{\theta}$ could be implicit in the set of equations $\mathrm{E}_{\mathbf{w}} \left[\mathbf{T}(\mathbf{w}, \mathbf{y}) | \mathbf{y}, \boldsymbol{\theta} \right] = \mathrm{E}_{\mathbf{y},\mathbf{w}} \left[\mathbf{T}(\mathbf{y}, \mathbf{w}) | \boldsymbol{\theta} \right]$. One way around this is to employ an iterative procedure, as detailed in Algorithm 3.

Algorithm 3 Exponential family EM

alg:EM3

1: **initialise** $\boldsymbol{\theta}^{(0)}$ and $t \leftarrow 0$
2: **while** not converged **do**
3: E-step: $\tilde{\mathbf{T}}^{(t+1)}(\mathbf{y}, \mathbf{w}) \leftarrow \mathrm{E}_{\mathbf{w}} \left[\mathbf{T}(\mathbf{w}, \mathbf{y}) | \mathbf{y}, \boldsymbol{\theta}^{(t)} \right]$
4: M-step: $\boldsymbol{\theta}^{(t+1)} \leftarrow$ solution to $\tilde{\mathbf{T}}^{(t+1)}(\mathbf{y}, \mathbf{w}) = \mathrm{E}_{\mathbf{y},\mathbf{w}} \left[\mathbf{T}(\mathbf{y}, \mathbf{w}) | \boldsymbol{\theta} \right]$
5: $t \leftarrow t + 1$
6: **end while**

To see how Algorithm 3 motivates the EM algorithm, consider the following argument. Recall that for the EM algorithm, the function $Q_t(\boldsymbol{\eta}) = \mathrm{E}_{\mathbf{w}}[\log p(\mathbf{y}, \mathbf{w}|\boldsymbol{\eta}) | \mathbf{y}, \boldsymbol{\eta}^{(t)}]$ is maximised at each iteration t. For exponential families of the form (4.21), the Q_t

function turns out to be

$$Q_t(\boldsymbol{\eta}) = \mathrm{E}_{\mathbf{w}}\left[\langle \boldsymbol{\eta}, \mathbf{T}(\mathbf{z})\rangle | \mathbf{y}, \boldsymbol{\eta}^{(t)}\right] - A^*(\boldsymbol{\eta}) + \log B(\mathbf{z}),$$

and this is maximised at the value of $\boldsymbol{\eta}$ satisfying

$$\frac{\partial}{\partial \boldsymbol{\eta}} Q_t(\boldsymbol{\eta}) = \mathrm{E}_{\mathbf{w}}\left[\mathbf{T}(\mathbf{y}, \mathbf{w}) | \mathbf{y}, \boldsymbol{\eta}^{(t)}\right] - \frac{\partial}{\partial \boldsymbol{\eta}} A^*(\boldsymbol{\eta}) = 0,$$

a similar condition to (4.23) when obtaining ML estimate of $\boldsymbol{\eta}$. Thus, Q_t is maximised by the solution to line 4 in Algorithm Algorithm 3.

4.10 Deriving the posterior predictive distribution

apx:postpred

A priori, assume that $y_{\text{new}} \sim \mathrm{N}(\hat{\alpha}, v_{\text{new}})$, where $v_{\text{new}} = \mathbf{h}_{\hat{\eta}}(x_{\text{new}})^\top \hat{\boldsymbol{\Psi}} \mathbf{h}_{\hat{\eta}}(x_{\text{new}}) + \psi_{\text{new}}^{-1}$. Consider the joint distribution of $(y_{\text{new}}, \mathbf{y}^\top)^\top$, which is multivariate normal (since both y_{new} and $\mathbf{y}$ are. Write

$$\begin{pmatrix} y_{\text{new}} \\ \mathbf{y} \end{pmatrix} \sim \mathrm{N}_{n+1}\left(\begin{pmatrix} \hat{\alpha} \\ \hat{\alpha}\mathbf{1}_n \end{pmatrix}, \begin{pmatrix} v_{\text{new}} & \mathrm{Cov}(y_{\text{new}}, \mathbf{y}) \\ \mathrm{Cov}(y_{\text{new}}, \mathbf{y})^\top & \tilde{\mathbf{V}}_y \end{pmatrix} \right),$$

where

$$\begin{aligned} \mathrm{Cov}(y_{\text{new}}, \mathbf{y}) &= \mathrm{Cov}(f_{\text{new}} + \epsilon_{\text{new}}, \mathbf{f} + \boldsymbol{\epsilon}) \\ &= \mathrm{Cov}(f_{\text{new}}, \mathbf{f}) + \mathrm{Cov}(\epsilon_{\text{new}}, \boldsymbol{\epsilon}) \\ &= \mathrm{Cov}\left(\mathbf{h}_{\hat{\eta}}(x_{\text{new}})^\top \tilde{\mathbf{w}}, \mathbf{H}_{\tilde{\eta}} \tilde{\mathbf{w}}\right) + (\sigma_{\text{new},1}, \ldots, \sigma_{\text{new},n}) \\ &= \mathbf{h}_{\hat{\eta}}(x_{\text{new}})^\top \hat{\boldsymbol{\Psi}} \mathbf{H}_{\tilde{\eta}} + \boldsymbol{\sigma}_{\text{new}}. \end{aligned}$$

The vector of covariances $\boldsymbol{\sigma}_{\text{new}}$ between observations $y_1, \ldots, y_n$ and the predicted point y_{new} would need to be prescribed a priori (treated as extra parameters), or estimated again, which seems excessive. Assuming $\boldsymbol{\sigma}_{\text{new}} = \mathbf{0}$ would be acceptable, especially under an iid assumption the error precisions. In any case, using standard multivariate normal

results, we get that $y_{\text{new}}|\mathbf{y}$ is also normally distributed with mean

$$\begin{aligned} E[y_{\text{new}}|\mathbf{y}] &= \hat{\alpha} + (\mathbf{h}_{\hat{\eta}}(x_{\text{new}})^\top \hat{\boldsymbol{\Psi}}\mathbf{H}_{\tilde{\eta}} + \boldsymbol{\sigma}_{\text{new}})\tilde{\mathbf{V}}_y^{-1}\tilde{\mathbf{y}} \\ &= \hat{\alpha} + \mathbf{h}_{\hat{\eta}}(x_{\text{new}})^\top \overbrace{\mathbf{h}_{\hat{\eta}}(x_{\text{new}})^\top \hat{\boldsymbol{\Psi}}\mathbf{H}_{\tilde{\eta}}\tilde{\mathbf{V}}_y^{-1}\tilde{\mathbf{y}}}^{\hat{\mathbf{w}}} + \boldsymbol{\sigma}_{\text{new}}\tilde{\mathbf{V}}_y^{-1}\tilde{\mathbf{y}} \\ &= \hat{\alpha} + \mathrm{E}\left[f(x_{\text{new}})|\mathbf{y}\right] + \text{mean correction term} \end{aligned}$$

and variance

$$\begin{aligned} \mathrm{Var}[y_{\text{new}}|\mathbf{y}] &= v_{\text{new}} - (\mathbf{h}_{\hat{\eta}}(x_{\text{new}})^\top \hat{\boldsymbol{\Psi}}\mathbf{H}_{\tilde{\eta}} + \boldsymbol{\sigma}_{\text{new}})\tilde{\mathbf{V}}_y^{-1}(\mathbf{h}_{\hat{\eta}}(x_{\text{new}})^\top \hat{\boldsymbol{\Psi}}\mathbf{H}_{\tilde{\eta}} + \boldsymbol{\sigma}_{\text{new}})^\top \\ &= \mathbf{h}_{\hat{\eta}}(x_{\text{new}})^\top \hat{\boldsymbol{\Psi}}\hat{\mathbf{h}}_{\hat{\eta}}(x_{\text{new}}) + \psi_{\text{new}}^{-1} - \mathbf{h}_{\hat{\eta}}(x_{\text{new}})^\top \hat{\boldsymbol{\Psi}}\mathbf{H}_{\tilde{\eta}}\tilde{\mathbf{V}}_y^{-1}\mathbf{H}_{\tilde{\eta}}\hat{\boldsymbol{\Psi}}\mathbf{h}_{\hat{\eta}}(x_{\text{new}}) \\ &\quad + \text{variance correction term} \\ &= \mathbf{h}_{\hat{\eta}}(x_{\text{new}})^\top \big(\hat{\boldsymbol{\Psi}} - \hat{\boldsymbol{\Psi}}\mathbf{H}_{\tilde{\eta}}\tilde{\mathbf{V}}_y^{-1}\mathbf{H}_{\tilde{\eta}}\hat{\boldsymbol{\Psi}}\big)\mathbf{h}_{\hat{\eta}}(x_{\text{new}}) + \psi_{\text{new}}^{-1} \\ &\quad + \text{variance correction term} \\ &= \mathbf{h}_{\hat{\eta}}(x_{\text{new}})^\top \hat{\mathbf{V}}_y^{-1}\mathbf{h}_{\hat{\eta}}(x_{\text{new}}) + \psi_{\text{new}}^{-1} + \text{variance correction term} \\ &= \mathrm{Var}\left[f(x_{\text{new}})|\mathbf{y}\right] + \psi_{\text{new}}^{-1} + \text{variance correction term.} \end{aligned}$$

4.11 Derivation of the Fisher information for multivariate normal distributions

apx:fisherm ultinormal

Let $X \sim \mathrm{N}_p(0, \Sigma_\theta)$, that is, the covariance matrix Σ_θ depends on a real, q-dimensional vector θ. Define the derivative of a matrix $\Sigma \in \mathbb{R}^{p\times p}$ with respect to a scalar z, denoted $\partial\Sigma/\partial z \in \mathbb{R}^{p\times p}$, by $(\partial\Sigma/\partial z)_{ij} = \partial\Sigma_{ij}/\partial z$, i.e. derivatives are taken elementwise. The two identities below are useful:

$$\frac{\partial}{\partial z}\operatorname{tr}\Sigma = \operatorname{tr}\frac{\partial\Sigma}{\partial z} \tag{4.24}$$

$$\frac{\partial}{\partial z}\log|\Sigma| = \operatorname{tr}\left(\Sigma^{-1}\frac{\partial\Sigma}{\partial z}\right) \tag{4.25}$$

$$\frac{\partial\Sigma^{-1}}{\partial z} = -\Sigma^{-1}\frac{\partial\Sigma}{\partial z}\Sigma^{-1} \tag{4.26}$$

A useful reference for these identities is Petersen, Pedersen, et al. (2008).

Taking derivative of the log-likelihood for θ with respect to the i'th component yields

$$\begin{aligned}\frac{\partial}{\partial\theta_i}L(\theta|X) &= -\frac{1}{2}\frac{\partial}{\partial\theta_i}\log|\Sigma_\theta| - \frac{1}{2}\frac{\partial}{\partial\theta_i}\operatorname{tr}(\Sigma_\theta^{-1}XX^\top) \\ &= -\frac{1}{2}\operatorname{tr}\left(\Sigma_\theta^{-1}\frac{\partial\Sigma_\theta}{\partial\theta_i}\right) - \frac{1}{2}\operatorname{tr}\left(\frac{\partial\Sigma_\theta^{-1}}{\partial\theta_i}XX^\top\right) \\ &= -\frac{1}{2}\operatorname{tr}\left(\Sigma_\theta^{-1}\frac{\partial\Sigma_\theta}{\partial\theta_i} - \Sigma_\theta^{-1}\frac{\partial\Sigma_\theta}{\partial\theta_i}\Sigma_\theta^{-1}XX^\top\right).\end{aligned}$$

Taking derivatives again, this time with respect to θ_j, we get

$$\begin{aligned}\frac{\partial^2}{\partial\theta_i\theta_j}L(\theta|X) &= -\frac{1}{2}\frac{\partial}{\partial\theta_j}\operatorname{tr}\left(\Sigma_\theta^{-1}\frac{\partial\Sigma_\theta}{\partial\theta_i} - \Sigma_\theta^{-1}\frac{\partial\Sigma_\theta}{\partial\theta_i}\Sigma_\theta^{-1}XX^\top\right) \\ &= -\frac{1}{2}\operatorname{tr}\Bigg(\frac{\partial\Sigma_\theta^{-1}}{\partial\theta_j}\frac{\partial\Sigma_\theta}{\partial\theta_i} + \Sigma_\theta^{-1}\frac{\partial^2\Sigma_\theta}{\partial\theta_i\theta_j} - \frac{\partial\Sigma_\theta^{-1}}{\partial\theta_j}\frac{\partial\Sigma_\theta}{\partial\theta_i}\Sigma_\theta^{-1}XX^\top \\ &\qquad - \Sigma_\theta^{-1}\frac{\partial^2\Sigma_\theta}{\partial\theta_i\theta_j}\Sigma_\theta^{-1}XX^\top - \Sigma_\theta^{-1}\frac{\partial\Sigma_\theta}{\partial\theta_i}\frac{\partial\Sigma_\theta^{-1}}{\partial\theta_j}XX^\top\Bigg).\end{aligned}$$

The Fisher information matrix U contains (i,j) entries equal to the expectation of $-\frac{\partial^2}{\partial\theta_i\theta_j}L(\theta|X)$. Using the fact that 1) $\mathrm{E}[\operatorname{tr}\Sigma] = \operatorname{tr}(\mathrm{E}\,\Sigma)$, 2) $\mathrm{E}[XX^\top] = \Sigma_\theta$; and 3) the trace is invariant under cyclic permutations, we get

$$\begin{aligned}U_{ij} &= \frac{1}{2}\operatorname{tr}\left(\cancel{\frac{\partial\Sigma_\theta^{-1}}{\partial\theta_j}\frac{\partial\Sigma_\theta}{\partial\theta_i}} + \cancel{\Sigma_\theta^{-1}\frac{\partial^2\Sigma_\theta}{\partial\theta_i\theta_j}} - \cancel{\frac{\partial\Sigma_\theta^{-1}}{\partial\theta_j}\frac{\partial\Sigma_\theta}{\partial\theta_i}} - \cancel{\Sigma_\theta^{-1}\frac{\partial^2\Sigma_\theta}{\partial\theta_i\theta_j}} - \frac{\partial\Sigma_\theta}{\partial\theta_i}\frac{\partial\Sigma_\theta^{-1}}{\partial\theta_j}\right) \\ &= \frac{1}{2}\operatorname{tr}\left(\Sigma_\theta^{-1}\frac{\partial\Sigma_\theta}{\partial\theta_i}\Sigma_\theta^{-1}\frac{\partial\Sigma_\theta}{\partial\theta_j}\right)\end{aligned}$$

as required.

Bibliography

bergsma2017 Bergsma, Wicher (2017). "Regression with I-priors". In: *Unpublished manuscript*.

bernardo2003variational Bernardo, JM, MJ Bayarri, JO Berger, AP Dawid, D Heckerman, AFM Smith, M West, et al. (2003). "The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures". In: *Bayesian statistics* 7, pp. 453–464.

carpenter2016stan Carpenter, Bob, Andrew Gelman, Matthew Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell (2017). "Stan: A Probabilistic Programming Language". In: *Journal of Statistical Software, Articles* 76.1, pp. 1–32. DOI: 10.18637/jss.v076.i01.

casella2002statistical Casella, George and Roger L Berger (2002). *Statistical inference*. Vol. 2. Duxbury Pacific Grove, CA.

chen2011single Chen, Dong, Peter Hall, and Hans-Georg Müller (2011). "Single and Multiple Index Functional Regression Models with Nonparametric Link". In: *The Annals of Statistics* 39.3, pp. 1720–1747. DOI: 10.1214/11-AOS882.

cheng2017variational Cheng, Ching-An and Byron Boots (2017). "Variational Inference for Gaussian Process Models with Linear Complexity". In: *Advances in Neural Information Processing Systems*, pp. 5190–5200.

davidian1995nonlinear Davidian, Marie and David M Giltinan (1995). *Nonlinear Models for Repeated Measurement Data*. Chapman and Hall/CRC.

dempster1977maximum Dempster, Arthur P, Nan M Laird, and Donald B Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38.

denwood2016runjags
Denwood, Matthew (2016). "**runjags**: An R Package Providing Interface Utilities, Model Templates, Parallel Computing Methods and Additional Distributions for MCMC Models in JAGS". In: *Journal of Statistical Software* 71.9, pp. 1–25. DOI: 10.18637/jss.v071.i09.

duane1987hybrid
Duane, Simon, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth (1987). "Hybrid monte carlo". In: *Physics letters B* 195.2, pp. 216–222.

eddelbuettel2011rcpp
Eddelbuettel, Dirk and Romain Francois (2011). "**Rcpp**: Seamless R and C++ Integration". In: *Journal of Statistical Software* 40.8, pp. 1–18. DOI: 10.18637/jss.v040.i08.

ferraty2006nonparametric
Ferraty, Frédéric and Philippe Vieu (2006). *Nonparametric Functional Data Analysis*. 1st. Springer-Verlag. DOI: 10.1007/0-387-36620-2.

fowlkes2001efficient
Fowlkes, C, S Belongie, and J Malik (2001). "Efficient Spatiotemporal Grouping Using the Nyström Method". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*. Vol. 1, pp. 231–238. DOI: 10.1109/CVPR.2001.990481.

hensman2013gaussian
Hensman, James, Nicolo Fusi, and Neil D Lawrence (2013). "Gaussian processes for big data". In: *arXiv preprint arXiv:1309.6835*.

jamil2017
Jamil, Haziq and Wicher Bergsma (2017). "iprior: An R Package for Regression Modelling using I-priors". In: *Manuscript in submission*.

kenward1987method
Kenward, Michael G. (1987). "A Method for Comparing Profiles of Repeated Measurements". In: *Journal of the Royal Statistical Society C (Applied Statistics)* 36.3, pp. 296–308. DOI: 10.2307/2347788.

caret
Kuhn, Max et al. (2017). **caret**: *Classification and Regression Training*. R package version 6.0–77. URL: https://CRAN.R-project.org/package=caret.

lange1995quasi
Lange, Kenneth (1995). "A quasi-Newton acceleration of the EM algorithm". In: *Statistica sinica*, pp. 1–18.

lian2014series
Lian, Heng and Gaorong Li (2014). "Series Expansion for Functional Sufficient Dimension Reduction". In: *Journal of Multivariate Analysis* 124.C, pp. 150–165. DOI: 10.1016/j.jmva.2013.10.019.

liu1998parameter
Liu, Chuanhai, Donald B Rubin, and Ying Nian Wu (1998). "Parameter expansion to accelerate EM: The PX-EM algorithm". In: *Biometrika* 85.4, pp. 755–770.

lunn2000winbugs
Lunn, David J., Andrew Thomas, Nicky Best, and David Spiegelhalter (Oct. 2000). "WinBUGS - A Bayesian modelling framework: Concepts, structure, and extensibility". In: *Statistics and Computing* 10.4, pp. 325–337. DOI: 10.1023/A:1008929526011.

meng1993maximum
Meng, Xiao-Li and Donald B Rubin (1993). "Maximum likelihood estimation via the ECM algorithm: A general framework". In: *Biometrika* 80.2, pp. 267–278.

neal2011mcmc
Neal, Radford M et al. (2011). "MCMC using Hamiltonian dynamics". In: *Handbook of Markov Chain Monte Carlo* 2.11.

jmcm
Pan, Jianxin and Yi Pan (2016). **jmcm**: *Joint Mean-Covariance Models using Armadillo and S4*. R package version 0.1.7.0. URL: https://CRAN.R-project.org/package=jmcm.

petersen2008matrix
Petersen, Kaare Brandt, Michael Syskind Pedersen, et al. (2008). "The matrix cookbook". In: *Technical University of Denmark* 7.15, p. 510.

pinheiro2000mixed
Pinheiro, José C and Douglas M Bates (2000). *Mixed-Effects Models in S and S-plus*. Springer-Verlag. DOI: 10.1007/b98882.

nlme
Pinheiro, Joséo, Douglas Bates, Saikat DebRoy, Deepayan Sarkar, and R Core Team (2017). **nlme**: *Linear and Nonlinear Mixed Effects Models*. R package version 3.1-131. URL: https://CRAN.R-project.org/package=nlme.

plummer2003jags
Plummer, Martyn (2003). "JAGS: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling". In: *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*. Vol. 124. Vienna, Austria, p. 125.

quinonero2005unifying
Quiñonero-Candela, Joaquin and Carl Edward Rasmussen (Dec. 2005). "A Unifying View of Sparse Approximate Gaussian Process Regression". In: *Journal of Machine Learning Research* 6, pp. 1939–1959.

rasmussen2006gaussian
Rasmussen, Carl Edward and Christopher K I Williams (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

rstan
Stan Development Team (2016). **RStan**: *The R Interface to Stan*. R package version 2.14.1. URL: http://mc-stan.org/.

sturtz2005r2winbugs
Sturtz, Sibylle, Uwe Ligges, and Andrew Gelman (2005). "**R2WinBUGS**: A Package for Running WinBUGS from R". In: *Journal of Statistical Software* 12.3, pp. 1–16. DOI: 10.18637/jss.v012.i03.

thodberg1996review Thodberg, Hans Henrik (1996). "A Review of Bayesian Neural Networks with an Application to near Infrared Spectroscopy". In: *IEEE Transactions on Neural Networks* 7.1, pp. 56–72. DOI: 10.1109/72.478392.

titsias2009variational Titsias, Michalis (2009). "Variational learning of inducing variables in sparse Gaussian processes". In: *Artificial Intelligence and Statistics*, pp. 567–574.

van2008reproducing van der Vaart, Aad W and van Zanten (2008). "Reproducing kernel Hilbert spaces of Gaussian priors". In: *Pushing the limits of contemporary statistics: contributions in honor of Jayanta K. Ghosh*. Institute of Mathematical Statistics, pp. 200–222.

wahba1990spline Wahba, Grace (1990). *Spline models for observational data*. Vol. 59. Siam.

williams2001using Williams, Christopher K I and Matthias Seeger (2001). "Using the Nyström Method to Speed Up Kernel Machines". In: *Advances in Neural Information Processing Systems 13*. The MIT Press, pp. 682–688.

yu2012monotonically Yu, Yaming (2012). "Monotonically overrelaxed EM algorithms". In: *Journal of Computational and Graphical Statistics* 21.2, pp. 518–537.

zellner1986assessing Zellner, Arnold (1986). "On assessing prior distributions and Bayesian regression analysis with g-prior distributions". In: *Bayesian inference and decision techniques*.

zhu2014structured Zhu, Hongxiao, Fang Yao, and Hao Helen Zhang (2014). "Structured Functional Additive Regression in Reproducing Kernel Hilbert Spaces". In: *Journal of the Royal Statistical Society B (Statistical Methodology)* 76.3, pp. 581–603. DOI: 10.1111/rssb.12036.