To-do list

1. Section 4.X 2
2. Section X 6
3. Running example: smoothing in one dimension. Data simulated, what are the true parameters? Run three methods of estimation, compare solutions, bias, MSE of prediction. 8
4. reference 8
5. Prove this in the appendix. IDEA is this: For MLE of exponential family, equate $T(x) = A'(\eta)$ and solve for η. Fact: $A'(\eta) = ET(x)$. Under EM, equate $E_zT(x,z) = ET(x,z)$, but $ET(x,z) = T(x,z)$ also if it satisfies MLE. 19

Contents

Haziq Jamil
Department of Statistics
London School of Economics and Political Science
March 18, 2018

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. 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 connect them to an appropriate RKKS, so that an I-prior may be defined on it. Methods for estimating I-prior models will also be described. Finally, several examples of I-prior modelling are presented.

4.1 Various regression models

4.2 Estimation

Depending on the type of regression model, an appropriate function space $\mathcal{F}$ needs to be chosen, and the reproducing kernel for the function space identified. Using an I-prior on the regression function $f \in \mathcal{F}$ with prior mean $f_0 \in \mathcal{F}$, the interest is then to obtain a posterior estimate of f. We denote the dependence of the kernel h on the parameters η

by h_η. The regression model to be estimated is always of the form

$$\begin{gathered}
y_i = \alpha + f_0(x_i) + \overbrace{\sum_{k=1}^n h_\eta(x_i, x_k) w_k}^{f(x_i)} + \epsilon_i \\
(\epsilon_1, \ldots, \epsilon_n)^\top \sim \mathrm{N}_n(\mathbf{0}, \mathbf{\Psi}^{-1}) \\
(w_1, \ldots, w_n)^\top \sim \mathrm{N}_n(\mathbf{0}, \mathbf{\Psi}),
\end{gathered} \tag{4.1}$$

although the indices may need to be adjusted for the individual model at hand, especially when dealing with ANOVA RKKSs. The parameters of the I-prior model are collectively denoted by $\theta = \{\alpha, \eta, \mathbf{\Psi}\}$. 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. Under all of these normality assumptions, using standard multivariate normal results one finds that the posterior of $\mathbf{w} := (w_1, \ldots, w_n)^\top$ is $\mathbf{w}|\mathbf{y} \sim \mathrm{N}_n(\tilde{\mathbf{w}}, \tilde{\mathbf{V}}_w)$, where

$$\tilde{\mathbf{w}} = \mathbf{\Psi}\mathbf{H}_\eta\mathbf{V}_y^{-1}(\mathbf{y} - \alpha\mathbf{1}_n - \mathbf{f}_0) \quad \text{and} \quad \tilde{\mathbf{V}}_w = \big(\mathbf{H}_\eta\mathbf{\Psi}\mathbf{H}_\eta + \mathbf{\Psi}^{-1}\big)^{-1} = \mathbf{V}_y^{-1}, \tag{4.2}$$

where $\mathbf{f}_0 = \big(f_0(x_1), \ldots, f_0(x_n)\big)^\top$, $\mathbf{H}_\eta$ is the kernel matrix with (i,j) entries equal to $h_\eta(x_i, x_j)$, and $\mathbf{V}_y$ is the variance of the marginal distribution for $\mathbf{y} = (y_1, \ldots, y_n)$. See Appendix 4.7 for a derivation.

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 $\mathcal{F}_1, \ldots, \mathcal{F}_p$, then at the very least, there are p scale parameters $\lambda_1, \ldots, \lambda_p$ for each of the RKHSs. Depending on the RKHS used, there could be either kernel parameters that need to be optimised—the Hurst index for the fBm RKHS, the lengthscale for the SE RKHS, and the offset for the polynomial RKKS. Default settings for these parameters may be used, and if this is the case, only scale parameters need to be estimated, and the estimation procedure can be made more efficient as the kernel matrices need not be recomputed each time. This is explained in further detail in Section 4.X.

For simplicity, the following additional assumptions are imposed on the I-prior model (4.1):

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

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

A3. Assume identical and independent errors, $\mathbf{\Psi} = \psi \mathbf{I}_n$.

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.

The following subsections describe possible estimation procedures for the hyperparameters of the model. Implementation of these estimation procedures are done in R, mainly using the **iprior** package (Jamil and Bergsma, 2017).

4.2.1 The intercept and the prior mean

In most statistical models, an intercept is a necessary inclusion to aid interpretation. In the regression model stated in (1.1), a lack of an intercept would fail to account for the correct location of the regression function with respect to the y-axis. Further, when zero-mean functions are considered, the intercept serves as being the 'grand mean' value of the responses.

There are two ways of including an intercept in the I-prior model. One is by including the tensor sum of the RKHS of constant functions to $\mathcal{F}$, and the other is to simply treat the intercept as a parameter of the model to be estimated. 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. 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 completely different. In the former 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 (4.1), $\operatorname{E} y_i = \alpha + f_0(x_i)$ for all $i = 1, \dots, n$, so the maximum likelihood estimate for $\operatorname{E} y$ 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 $\mathbf{w}$, θ 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

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

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.

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

Assuming 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 (4.1), the log-likelihood of θ is given by

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

This is typically done using conjugate gradients with a Cholesky decomposition on the covariance kernel to maintain stability, but the **iprior** package opts for an eigendecomposition of the kernel matrix $\mathbf{H}_\eta = \mathbf{V}\cdot\mathrm{diag}(u_1,\dots,u_n)\cdot\mathbf{V}^\top$ instead. Further, assuming A3 and since $\mathbf{H}_\eta$ is a symmetrix matrix, we have that $\mathbf{V}\mathbf{V}^\top = \mathbf{I}_n$, and thus

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

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.

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

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

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 $\mathbf{U}^{-1}$. In particular, the standard errors for θ_k are the diagonal elements of $\mathbf{U}^{-1/2}$.

4.2.3 Expectation-maximisation algorithm

We describe an expectation-maximisation algorithm to estimate both the posterior regression function and the hyperparameters of (4.1) simultaneously. Assume A1 and A2 applies. Evidently, (4.1) lends itself to resembling a random-effects model. 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

$$\begin{aligned}
Q(\theta) &= \mathrm{E}_{\mathbf{w}}\left[\log p(\mathbf{y}, \mathbf{w}|\theta) \,\middle|\, \mathbf{y}, \theta^{(t)}\right] \\
&= \mathrm{E}_{\mathbf{w}}\left[\text{const.} - \frac{1}{2}(\tilde{\mathbf{y}} - \mathbf{H}_\eta \mathbf{w})^\top \mathbf{\Psi} (\tilde{\mathbf{y}} - \mathbf{H}_\eta \mathbf{w}) - \frac{1}{2}\mathbf{w}^\top \mathbf{\Psi}^{-1}\mathbf{w} \,\middle|\, \mathbf{y}, \theta^{(t)}\right] \qquad (4.3) \\
&= \text{const.} - \frac{1}{2}\tilde{\mathbf{y}}^\top \mathbf{\Psi}\tilde{\mathbf{y}} - \frac{1}{2}\operatorname{tr}\left(\overbrace{(\mathbf{H}_\eta \mathbf{\Psi} \mathbf{H}_\eta + \mathbf{\Psi}^{-1})}^{\Sigma_\theta}\tilde{\mathbf{W}}^{(t)}\right) + \tilde{\mathbf{y}}^\top \mathbf{\Psi}\mathbf{H}_\eta \tilde{\mathbf{w}}^{(t)},
\end{aligned}$$

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 (4.2), substituting $\theta^{(t)}$ as appropriate. Note that (4.3) follows as a direct consequence of the results in Appendix 4.7.

Assume that A3 applies. 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

$$\frac{\partial Q}{\partial \eta} = -\frac{1}{2}\operatorname{tr}\left(\frac{\partial \mathbf{\Sigma}_\theta}{\partial \eta}\tilde{\mathbf{W}}^{(t)}\right) + \psi \cdot \tilde{\mathbf{y}}^\top \frac{\partial \mathbf{H}_\eta}{\partial \eta}\tilde{\mathbf{w}}^{(t)} \qquad (4.4)$$

$$\frac{\partial Q}{\partial \psi} = -\frac{1}{2}\tilde{\mathbf{y}}^\top \tilde{\mathbf{y}} - \operatorname{tr}\left(\frac{\partial \mathbf{\Sigma}_\theta}{\partial \psi}\tilde{\mathbf{W}}^{(t)}\right) + \tilde{\mathbf{y}}^\top \mathbf{H}_\eta \tilde{\mathbf{w}}^{(t)} \qquad (4.5)$$

equated to zero. As $\partial \mathbf{\Sigma}_\theta / \partial \psi = \mathbf{H}_\eta^2 - \psi^{-2}$, the solution to (4.5) is obtained as

$$\psi^{(t+1)} = \left\{\frac{\operatorname{tr}\tilde{\mathbf{W}}^{(t)}}{\tilde{\mathbf{y}}^\top \tilde{\mathbf{y}} + \operatorname{tr}(\mathbf{H}_\eta^2 \tilde{\mathbf{W}}^{(t)}) - 2\tilde{\mathbf{y}}^\top \mathbf{H}_\eta \tilde{\mathbf{w}}^{(t)}}\right\}^{1/2}. \qquad (4.6)$$

The solution to (4.4) can also be found in closed-form, but only in cases where the full-data likelihood in η emits itself as belonging to an exponential family likelihood. Such cases are described in further detail in Section X. 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, EM algorithm is more stable than direct maximisation, in that it is less prone to the likelihood exploding should any of the parameters approach problematic boundary values. As such, the EM is especially suitable if there are many scale parameters to estimate. 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. Section X also describes various strategies to run the EM algorithm efficiently.

As a final remark, it is well known that the EM algorithm increases the value of the log-likelihood at each iteration. It is also known that the EM sequence $\theta^{(t)}$ eventually convergences to some θ^*, but the fact that θ^* is the ML estimate is not guaranteed. The paper by Wu (1983) details the conditions necessary for the EM to produce ML estimates. If the EM tends to get stuck in some local maxima of the likelihood, then a general strategy is to restart the EM from multiple starting values.

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. 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. Note that this full Bayesian method is not implemented in **iprior**, but described here for completeness.

4.2.5 Comparison of estimation methods

Running example: smoothing in one dimension. Data simulated, what are the true parameters? Run three methods of estimation, compare solutions, bias, MSE of prediction.

4.3 Computational considerations

Computational complexity for estimating I-prior models (and in fact, for GPR in general) is dominated by the inversion of the $n \times n$ matrix $\mathbf{\Sigma}_\theta = \mathbf{H}_\eta \mathbf{\Psi} \mathbf{H}_\eta + \mathbf{\Psi}^{-1}$, which scales as $O(n^3)$ in time. As mentioned earlier, the **iprior** package inverts this by way of the eigendecomposition of $\mathbf{H}_\eta$, but this operation is also $O(n^3)$. 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 use low-rank matrix approximations. 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

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

obtained via the Woodbury matrix identity, is a potentially 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 the kernel matrix itself is sufficiently low ranked (for instance, 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 reference 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 $\{1, \dots, n\}$, 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 $\tilde{\mathbf{v}}_i$ 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 $u_1^{(m)}, \dots, u_m^{(m)}$, 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 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 and $O(nm)$ storage.

There is the issue of selecting the subset $\mathcal{M}$. 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

The evaluation of the Q function in (4.3) 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 $\lambda = \{\lambda_1, \dots, \lambda_p\}$ and the iid error precision ψ under assumptions A1–A3.

Assume that any other kernel parameters are fixed and need not be estimated, and that there are p scale parameters corresponding to p RKHS $\mathcal{F}_1, \dots, \mathcal{F}_p$ of functions over $\mathcal{X}_1, \dots, \mathcal{X}_p$. 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 $\mathcal{X}$ itself may be more than one-dimensional. The kernel over $\mathcal{X} \times \mathcal{X}$ is therefore

$$h_\eta = \lambda_1 h_1.$$

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
$$h_\eta = \lambda_1 h_1 + \cdots + \lambda_p h_p.$$

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, and its kernel is
$$h_\eta = \sum_{j=1}^{p} \lambda_j h_j + \sum_{j<k} \lambda_j \lambda_k h_j h_k,$$
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 $(f - \alpha) \in \mathcal{F}$, then the kernel of $\mathcal{F}$ is
$$h_\eta = \sum_{k=1}^{d} b_k \lambda_1^k h_1^k.$$

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 $\lambda = \{\lambda_1, \dots, \lambda_p\}$ and the set of kernel functions $h = \{h_1, \dots, h_p\}$. Furthermore, scenarios 1–3 are special cases of the ANOVA RKKS excluding the grand mean[1]: in 1. and 2., $\mathcal{F}$ is the ANOVA RKKS of main effects only, and in 3., $\mathcal{F}$ is the ANOVA RKKS of main effects and level-two interactions.

Let us be more concrete about what we mean by 'manipulation' of the sets λ and h. Define an 'instruction' operator ι 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. For the four scenarios above, the list $\mathcal{Q}$ is

1. $\mathcal{Q} = \big\{\{1\}\big\}$.
2. $\mathcal{Q} = \big\{\{1\}, \dots, \{p\}\big\}$.

[1]As discussed, for simplicity the RKHS of constant functions is ignored and the model includes an intercept to be estimated instead.

3. $\mathcal{Q} = \big\{\{1\},\dots,\{p\},\{1,2\},\dots,\{p-1,p\}\big\}$.

4. $\mathcal{Q} = \big\{\{1\},\{1,1\},\dots,\{1,\dots,1\}\big\}$.

For the polynomial RKKS in the fourth example, then one must also multiply the constants b_k as appropriate. Let q be the cardinality of the set $\mathcal{Q}$, that is, the number of summands required to construct the kernel for $\mathcal{F}$. Denote the instructed sets as $\xi = \{\xi_1,\dots,\xi_q\}$ for λ and $a = \{a_1,\dots,a_q\}$ for h. Therefore, this allows us to write the kernel h_η as a linear combination of ξ and a,

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

The reason this is important is because changes in λ for h_η only changes the ξ_k's, but the instructed kernels $a_1,\dots,a_q$ do not. This allows us to compute and store all of the required $n \times n$ kernel matrices $\mathbf{A}_1,\dots,\mathbf{A}_q$ 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. The application of the instruction set $\mathcal{Q}$ to λ to obtain ξ is computationally effortless.

Notice that

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

Provided that we have the matrices $\mathbf{A}_{jk} = \mathbf{A}_j\mathbf{A}_k$, $j,k = 1,\dots,q$ in addition to $\mathbf{A}_1,\dots,\mathbf{A}_q$ pre-calculated and stored, then evaluating $\operatorname{tr}\big(\mathbf{A}_{jk}\tilde{\mathbf{W}}^{(t)}\big) = \operatorname{vec}(\mathbf{A}_{jk})^\top \operatorname{vec}(\tilde{\mathbf{W}}^{(t)})$ 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 Appendix 4.8 for details). Certain I-prior models emit this property, namely regression functions belonging to the full or non-full ANOVA RKKS, and we describe its estimation below.

Assume A1–A3 applies, and that only the error precision ψ and the RKHS scale parameters $\lambda_1, \dots, \lambda_p$ need to be estimated, i.e. all other kernel parameters are fixed. 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 as the terms of the ANOVA kernel involving λ_k, and the terms not involving λ_k, respectively. 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 $\mathcal{R}_k^\lambda$, 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

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

Consider the full data log-likelihood for λ_k, $k = 1,\dots,p$, conditionally dependent on the rest of the unknown parameters ψ and $\lambda_{-k} = \{\lambda_1,\dots,\lambda_p\}\backslash\{\lambda_k\}$:

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

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 $w_1,\dots,w_n$ 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 (4.2). That the λ_k's have closed-form expressions, together with the closed-form expression for ψ in (4.6), 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. This form of the EM algorithm is known as the *conditional expectation-maximisation* algorithm (Meng and Rubin, 1993).

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, or in fact dropping any of the terms in the ANOVA kernel, can be used by this method.

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

```
1:  procedure INITIALISATION
2:      Initialise $\lambda_1^{(0)}, \dots, \lambda_p^{(0)}, \psi^{(0)}$
3:      Compute and store matrices as per $\mathcal{R}_k$ and $\mathcal{S}_k$.
4:      $t \leftarrow 0$
5:  end procedure

6:  while not converged do
7:      procedure E-STEP
8:          $\tilde{\mathbf{w}} \leftarrow \psi^{(t)}\mathbf{H}_{\eta^{(t)}}\big(\psi^{(t)}\mathbf{H}_{\eta^{(t)}}^2 + \psi^{-(t)}\mathbf{I}_n\big)^{-1}\tilde{\mathbf{y}}$
9:          $\tilde{\mathbf{W}} \leftarrow \big(\psi^{(t)}\mathbf{H}_{\eta^{(t)}}^2 + \psi^{-(t)}\mathbf{I}_n\big)^{-1} + \tilde{\mathbf{w}}\tilde{\mathbf{w}}^\top$
10:     end procedure

11:     procedure M-STEP
12:         for $k = 1, \dots, p$ do
13:             $T_{1k} \leftarrow \frac{1}{2}\operatorname{tr}(\mathbf{R}_k^2\tilde{\mathbf{W}})$
14:             $T_{2k} \leftarrow \tilde{\mathbf{y}}^\top\mathbf{R}_k\tilde{\mathbf{w}} - \frac{1}{2}\operatorname{tr}(\mathbf{U}_k^2\tilde{\mathbf{W}}^\top)$
15:             $\lambda_k^{(t+1)} \leftarrow T_{2k}/2T_{1k}$
16:         end for
17:         $T_3 \leftarrow \tilde{\mathbf{y}}^\top\tilde{\mathbf{y}} + \operatorname{tr}(\mathbf{H}_{\eta^{(t)}}^2\tilde{\mathbf{W}}^{(t)}) - 2\tilde{\mathbf{y}}^\top\mathbf{H}_{\eta^{(t)}}\tilde{\mathbf{w}}^{(t)}$
18:         $\psi^{(t+1)} \leftarrow \operatorname{tr}\tilde{\mathbf{W}}^{(t)}/T_3$
19:     end procedure
20:     $t \leftarrow t + 1$
21: end while
```

While the exponential family EM algorithm takes similar computational time as the efficient EM algorithm described in (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, a similar conjugacy argument for the ψ parameter can be done, whereby the gamma distribution is the density in question. The usual 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 4.1. 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

Small speed up to the EM algorithm.

4.4 Post-estimation

4.5 Examples

4.6 Conclusion

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

1. Select an appropriate function space; equivalently, the kernels for which a specific effect is desired on the covariates. Several modelling examples are described in Section 4.1.
2. Estimate the hyperparameters (these included the RKHS scale parameter(s), error precision, and any other kernel parameters such as the Hurst index of fBm) of the I-prior model and obtain the posterior regression function.
3. Post-estimation procedures include
   - 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 Nyström method of approximating the kernel matrix reduces complexity to $O(nm)$ storage and approximately $O(nm^2)$, and is highly advantageous if $m \ll n$. 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.

One promising avenue to achieve efficient computation for I-prior models is by using variational methods. A sparse variational approximation (typically by using inducing points) or stochastic variational inference can greatly reduce computational storage and speed requirements. A recent paper by Cheng and Boots (2017) suggested a variational algorithm with linear complexity for GPR-type models.

Omitted

Appendix

4.7 Deriving the posterior distribution for w

In the following derivation, we implicitly assume the dependence on $\mathbf{f}_0$ and θ. The distribution of $\mathbf{y}|\mathbf{w}$ is $\text{N}_n(\boldsymbol{\alpha}+\mathbf{f}_0+\mathbf{H}_\eta\mathbf{w}, \boldsymbol{\Psi}^{-1})$, where $\boldsymbol{\alpha} = \alpha\mathbf{1}_n$, while the prior distribution for $\mathbf{w}$ is $\text{N}_n(\mathbf{0}, \boldsymbol{\Psi})$. 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

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

and that $\operatorname{Cov}(\mathbf{w}, \mathbf{y}) = \boldsymbol{\Psi} \mathbf{H}_\eta = \mathbf{H}_\eta \boldsymbol{\Psi} = \operatorname{Cov}(\mathbf{y}, \mathbf{w})$ 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,

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

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.

4.8 A recap on the exponential family EM algorithm

Consider the density function $p(\cdot|\boldsymbol{\theta})$ of the complete data $\mathbf{z} = \{\mathbf{y}, \mathbf{w}\}$, which depends on parameters $\boldsymbol{\theta} = (\theta_1, \dots, \theta_s)^\top \in \Theta \subseteq \mathbb{R}^s$, 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 $\boldsymbol{\eta} : \mathbb{R}^s \mapsto \mathbb{R}$, $\mathbf{T}(\mathbf{z}) = \big(T_1(\mathbf{z}), \dots, T_s(\mathbf{z})\big)^\top \in \mathbb{R}^s$ 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

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

by defining $\boldsymbol{\eta} := \big(\eta_1(\boldsymbol{\theta}), \dots, \eta_r(\boldsymbol{\theta})\big) \in \mathcal{E}$, 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 $\mathcal{E} := \big\{\boldsymbol{\eta} = (\eta_1, \dots, \eta_s) \,|\, \int \exp A^*(\boldsymbol{\eta}) < \infty \big\}$ 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

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

A useful identity to know is that $\frac{\partial}{\partial \boldsymbol{\eta}} A^*(\boldsymbol{\eta}) = \mathrm{E}\, \mathbf{T}(\mathbf{z})$ (Casella and R. L. Berger, 2002, Theorem 3.4.2 & Exercise 3.32(a)).

Prove this in the appendix. IDEA is this: For MLE of exponential family, equate $T(x) = A'(\eta)$ and solve for η. Fact: $A'(\eta) = ET(x)$. Under EM, equate $E_z T(x, z) = ET(x, z)$, but $ET(x, z) = T(x, z)$ also if it satisfies MLE.

$\boldsymbol{\eta} = \big(\eta_1(\boldsymbol{\theta}), \dots, \eta_1(\boldsymbol{\theta})\big) \in \mathbb{R}^s$

Bibliography

Bernardo, J., M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. 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.

Carpenter, B., A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell (2017). “Stan: A Probabilistic Programming Language”. In: *Journal of Statistical Software, Articles* 76.1, pp. 1–32. DOI: 10.18637/jss.v076.i01.

Casella, G. and R. L. Berger (2002). *Statistical inference.* Vol. 2. Duxbury Pacific Grove, CA.

Cheng, C.-A. and B. Boots (2017). “Variational Inference for Gaussian Process Models with Linear Complexity”. In: *Advances in Neural Information Processing Systems*, pp. 5190–5200.

Dempster, A. P., N. M. Laird, and D. 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.

Denwood, M. (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.

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

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`.

Jamil, H. and W. Bergsma (2017). “iprior: An R Package for Regression Modelling using I-priors”. In: *Manuscript in submission.*

Lunn, D. J., A. Thomas, N. Best, and D. 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`.

Meng, X.-L. and D. B. Rubin (1993). “Maximum likelihood estimation via the ECM algorithm: A general framework”. In: *Biometrika* 80.2, pp. 267–278.

Plummer, M. (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.

Quiñonero-Candela, J. and C. E. Rasmussen (Dec. 2005). “A Unifying View of Sparse Approximate Gaussian Process Regression”. In: *Journal of Machine Learning Research* 6, pp. 1939–1959.

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

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

Sturtz, S., U. Ligges, and A. 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`.

Williams, C. K. I. and M. 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.

Wu, C. J. (1983). “On the convergence properties of the EM algorithm”. In: *The Annals of statistics*, pp. 95–103.