To-do list

Contents

Haziq Jamil
Department of Statistics
London School of Economics and Political Science
March 13, 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

After choosing an appropriate function space... What are the kernel parameters? State the model $y = \alpha + \sum hw + e$. Goal is to estimate posterior regression function.

After imposing an I-prior on the regression function, the interest is then to obtain the posterior distribution of the regression function. This has been described in Chapter 1, and repeated here for convenience. In particular, the posterior distribution for the regression function of the form $f(x) = \sum_{i=1}^n h(x, x_i)w_i$ merely depends on the posterior distribution of $\mathbf{w} := (w_1, \dots, w_n)^\top$, which is $\mathbf{w}|\mathbf{y} \sim \mathrm{N}_n(\tilde{\mathbf{w}},)$, where

$$\tilde{\mathbf{w}} =$$

explain package

4.2.1 The intercept

Given the regression model (1.1) subject to an I-prior (1.2), the marginal likelihood of the intercept α (after integrating out the I-prior) can be maximised with respect to α, which yields the sample mean for y as the ML estimate for intercept.

4.2.2 Direct optimisation

The kernel parameter η and the error precision ψ (which we collectively refer to as the model hyperparameters of the covariance kernel θ) can be estimated in several ways. One of these is direct optimisation of the marginal log-likelihood—the most common method in the Gaussian process literature.

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

where $\mathbf{\Sigma}_\theta = \psi\mathbf{H}_\eta^2 + \psi^{-1}\mathbf{I}_n$. 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 (Gram matrix) $\mathbf{H}_\eta = \mathbf{V}\cdot\operatorname{diag}(u_1,\dots,u_n)\cdot\mathbf{V}^\top$ instead. Since $\mathbf{H}_\eta$ is a symmetrix matrix, we have that $\mathbf{V}\mathbf{V}^\top = \mathbf{I}_n$, and thus

$$\mathbf{\Sigma}_\theta = \mathbf{V}\cdot\operatorname{diag}(\psi u_1^2 + \psi^{-1},\dots,\psi u_n^2 + \psi^{-1})\cdot\mathbf{V}^\top$$

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.

4.2.3 Expectation-maximisation algorithm

Alternatively, the expectation-maximisation (EM) algorithm may be used to estimate the hyperparameters, in which case the I-prior formulation in (**??**) is convenient. Substituting this into (**??**) we get something that resembles a random effects model. By treating the w_i as "missing", the tth iteration of the E-step entails computing

$$Q(\theta) = \mathrm{E}\left[\log p(\mathbf{y}, \mathbf{w}|\theta) \big| \mathbf{y}, \theta^{(t)}\right]. \tag{4.1}$$

As a consequence of the properties of the normal distribution, the required joint and posterior distributions $p(\mathbf{y}, \mathbf{w})$ and $p(\mathbf{w}|\mathbf{y})$ are easily obtained. The M-step then maximises the Q function above, which boils down to solving the first order conditions

$$\frac{\partial Q}{\partial \eta} = -\frac{1}{2}\operatorname{tr}\left(\frac{\partial \mathbf{\Sigma}_\theta}{\partial \eta}\tilde{\mathbf{W}}^{(t)}\right) + \psi \cdot \mathbf{y}^\top \frac{\partial \mathbf{H}_\eta}{\partial \eta}\tilde{\mathbf{w}}^{(t)} \tag{4.2}$$

$$\frac{\partial Q}{\partial \psi} = -\frac{1}{2}\mathbf{y}^\top\mathbf{y} - \operatorname{tr}\left(\frac{\partial \mathbf{\Sigma}_\theta}{\partial \psi}\tilde{\mathbf{W}}^{(t)}\right) + \mathbf{y}^\top \mathbf{H}_\eta \tilde{\mathbf{w}}^{(t)} \tag{4.3}$$

equated to zero. Here, $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{W}}$ are the first and second posterior moments of $\mathbf{w}$. The solution to (4.3) can be found in closed-form, but not necessarily for (4.2). In cases where closed-form solutions exist, 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 θ, the Q function is again optimised with respect to θ using the L-BFGS algorithm.

The EM algorithm is more stable than direct maximization, and is especially suitable if there are many scale parameters. However, 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 hyperparameters set. 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 $\mathbf{\Sigma}_\theta = \psi \mathbf{H}_\eta^2 + \psi^{-1}\mathbf{I}_n$, 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 $\tilde{\mathbf{w}}$, the posterior mean of the I-prior random effects. Furthermore, storage requirements for I-priors models are similar to that of GPR models, which is $O(n^2)$.

4.3.1 The Nyström approximation

The machine learning literature is rich in ways to resolve this issue, as summarised by Quiñonero-Candela and Rasmussen, 2005. One such method is to use low-rank matrix approximations. Let $\mathbf{Q}$ be a matrix with rank $q < n$, and that $\mathbf{Q}\mathbf{Q}^\top$ can be used to approximate 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 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. 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.

Another method of approximating the kernel matrix, and the method implemented by our package, 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. Reorder the rows and columns and partition the kernel matrix as

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

The Nyström method provides an approximation to the lower right block $\mathbf{C}$ by manipulating the eigenvectors and eigenvalues of $\mathbf{A}$, an $m \times m$ matrix, together with the matrix $\mathbf{B}$ to give

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

where $\mathbf{U}_m$ is the diagonal matrix containing the m eigenvalues of $\mathbf{A}$, and $\mathbf{V}_m$ is the corresponding matrix of eigenvectors. An orthogonal version of this approximation is of interest, which has been studied by Fowlkes et al., 2001, which allows us to easily calculate the inverse of $\mathbf{\Sigma}_\theta$. Estimating I-prior models using the Nyström method takes $O(nm^2)$ times and $O(nm)$ storage.

4.3.2 An efficient EM algorithm

As a sacrifice: store all kernel matrices beforehand. When lambdas are in closed form... what is the computational time complexity? Describe lambda in closed form. What happens if lambda not in closed form. psi always in closed form.

4.3.3 Convex EM

Small speed up to the EM algorithm. Adaptive?

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

Bibliography

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.

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.

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.

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.

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.

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.

List of Figures

List of Tables

List of Theorems

List of Definitions