APTS Statistical Computing Assessment 2012

The work provided here is intended to take up to half a week to complete. Students should talk to their supervisors to find out whether or not their department requires this work as part of any formal accreditation process (APTS itself has no resources to assess or certify students). It is anticipated that departments will decide on the appropriate *level* of assessment locally, and may choose to drop some (or indeed all) of the parts, accordingly. So make sure that your supervisor or local organizer of APTS assessment has looked at the assignment before you start, and has told you which parts of it to do. In order to avoid undermining institutions' local assessment procedures the module lecturer will not respond to enquiries from students about this assignment.

Consider the simple smooth additive regression model

$$y_i = \alpha + \sum_j f_j(x_{ji}) + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2) \tag{1}$$

where y_i is a univariate response variable, x_{ji} is the i^{th} observation of the j^{th} predictor variable, and the f_j are unknown smooth functions, which are to be estimated. A convenient way of estimating such models is by penalized least squares. The f_j are each represented by some linear basis expansion, and are each subjected to a tuneable penalty during fitting, where the strength of penalization controls the smoothness of the estimate of f_j. (See library `mgcv` in R, for example.)

Ruppert, Wand and Carroll (2003) advocate a particularly straightforward basis and penalty, based on the 'truncated power basis' for splines. Dropping the index j to avoid clutter, and supposing that x_k^* are K values of x, evenly spaced through the range of the observed x_i values, then let

$$f(x) = \sum_{i=1}^{m} x^i \gamma_i + \sum_{k=1}^{K} \beta_k (x - x_k^*)_+^m$$

where m is a small integer (usually 1,2 or 3) and $(z)_+ = z$ if $z > 0$ and 0 otherwise. γ_i and β_k are parameters to be estimated and $f(x)$ has no intercept, since this would only be confounded with α. RWC suggest using $\lambda \beta^{\mathsf{T}} \beta$ as a penalty measuring the wiggliness of f, where λ is a 'smoothing parameter' used to tune the strength of the penalty during fitting.

Using such a basis expansion for each f_j, then (1) can be re-written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$$

where $\mathbf{X}$ is a model matrix implied by the basis expansion, and $\boldsymbol{\beta}$ is a vector containing all the unknown model coefficients (i.e. $(\alpha, \gamma_{11}, \gamma_{12}, \beta_{11}, \beta_{12}, \ldots, \ldots, \beta_{1K}, \gamma_{21}, \gamma_{22}, \beta_{21}, \beta_{22}, \ldots, \ldots, \beta_{2K}, \ldots)^{\mathsf{T}}$, or similar). The overall wiggliness[1] penalty for the model is then the sum of the individual wiggliness penalties, and can be written, $\boldsymbol{\beta}^{\mathsf{T}} \mathbf{S}_\lambda \boldsymbol{\beta}$, where $\mathbf{S}_\lambda$ is a diagonal matrix with a leading diagonal something like $(0, 0, 0, \sqrt{\lambda_1}, \ldots, \sqrt{\lambda_1}, 0, 0, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_2}, \ldots)$ (the blocks of $\sqrt{\lambda_j}$ being each of length K, while the 0 blocks are of length m, except for the first which of length $m + 1$ since α is unpenalized).

Given this representation, and some values for the smoothing parameter, the model can be fitted by minimizing

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S}_\lambda \boldsymbol{\beta}$$

w.r.t. $\boldsymbol{\beta}$, which has the formal solution $\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathsf{T}}\mathbf{X} + \mathbf{S}_\lambda)^{-1} \mathbf{X}^{\mathsf{T}} \mathbf{y}$. Smoothing parameters can be estimated by minimizing the generalized cross validation score

$$\mathcal{V}(\lambda) = n \|\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2 / (n - \text{tr}(\mathbf{A}))^2$$

where n is the dimension of $\mathbf{y}$ and $\mathbf{A} = \mathbf{X}(\mathbf{X}^{\mathsf{T}}\mathbf{X} + \mathbf{S}_\lambda)^{-1}\mathbf{X}^{\mathsf{T}}$. A covariance matrix for the parameters is $(\mathbf{X}^{\mathsf{T}}\mathbf{X} + \mathbf{S}_\lambda)^{-1}\sigma^2$ and $\hat{\sigma}^2 = \|\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2 / (n - \text{tr}(\mathbf{A}))$.

[1] rugged men of action types prefer the less descriptive term 'roughness penalty' for some reason (probably related to enjoying Hemingway, smoking Malborough etc.)

1. Write an R routine called `am.setup`, which has arguments `x`, `k` and `m`. `x` is a matrix each column of which contains the observations for one predictor variable. `k` is the number of knots to use, and should default to 10, `m` should default to 2. The routine should create the model matrix for a model of the form (1).

   In the interests of numerical stability, each predictor variable should be centred, by subtracting its mean, before setting up the bases. The routine should create knots x_k^* for each smooth itself. Make sure that the last knot is before the last x observation, otherwise the last basis function will not actually be used. Your routine should return a list containing the model matrix, the shifts applied to the predictors, `k`, `m`, and a matrix containing the knots. (See `?colMeans` `?sweep` & `?sign`.)

2. By excluding constant terms from the basis representations of the f_j we ensured that they will be formally identifiable, but to obtain the smallest possible confidence intervals for the f_j, we would actually like the bases to be orthogonal to the intercept. This is easily done by forcing each column of $\mathbf{X}$, except the intercept column, to average to zero. To do this, modify your code so that all the columns of $\mathbf{X}$, except the first, are transformed by subtraction of their mean. Include the vector of subtracted column means as an extra item in the list returned by `am.setup`.

3. Test your code by fitting the model $\texttt{accel}_i = \alpha + f(\texttt{times}_i) + \epsilon_i$ to the `motorcycle` data from the `MASS` library in R, without penalization (you can just use `lm` for fitting here).

4. Now write a routine `am.penalty` which take arguments `sp`, `k` and `m` and returns $\mathbf{S}_\lambda$, if `sp` contains the vector of smoothing parameters $\boldsymbol{\lambda}$. Notice that your routine automatically returns $\sqrt{\mathbf{S}_\lambda}$ if `sp` contains the square roots of the smoothing parameters.

5. Consider the QR decompositions $\mathbf{X} = \mathbf{QR}$ and $\begin{pmatrix} \mathbf{R} \\ \sqrt{\mathbf{S}_\lambda} \end{pmatrix} = \mathbf{Q}_1\mathbf{R}_1$, and let $\mathbf{Q}_{11}$ denote the sub-matrix of $\mathbf{Q}_1$ such that $\mathbf{R} = \mathbf{Q}_{11}\mathbf{R}_1$. Show that $\hat{\boldsymbol{\beta}} = \mathbf{R}_1^{-1}\mathbf{Q}_{11}^\mathsf{T}\mathbf{Q}^\mathsf{T}\mathbf{y}$, and $\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{Q}_{11}\mathbf{Q}_{11}^\mathsf{T})$.

6. Modify `am.setup` to return the qr decomposition of $\mathbf{X}$ instead of $\mathbf{X}$. Now write a routine `am.fit` with arguments `sp`, `am` and `y`, where `sp` is a smoothing parameter vector (or square root smoothing parameter vector), `am` is the list returned from `am.setup` and `y` is the vector of response observations. Using the results from part 5, your routine should fit (1), returning, $\hat{\boldsymbol{\beta}}$, $\hat{\boldsymbol{\mu}}$ $(= \mathbf{X}\hat{\boldsymbol{\beta}})$, $\text{tr}(\mathbf{A})$ and the GCV score. (See `?qr` and `?qr.R` for useful routines.)

7. Now try fitting a penalized version of the motorcycle data model from part 3, using `k=20`. First experiment with different `sp` values to check that the estimates behave as expected as the smoothing parameter changes. Then write a loop to search for the smoothing parameter minimizing the GCV score over a grid of values. (A grid of 40 log square root smoothing parameters between -2 and 5 should be sufficient). Finally refit with the GCV optimal smoothing parameter and produce a plot of acceleration against time with the smoothed accelerations against time overlaid.

8. Write a function `am.predict` with arguments `am` and `x`. `am` is the list returned from `am.setup`, `x` is a matrix like that originally passed to `am.setup`, but containing new values for the predictor variables, at which new predictions are required. The function should return a prediction matrix, $\mathbf{X}^p$ such that $\mathbf{X}^p\hat{\boldsymbol{\beta}}$ gives the predicted values. Computation of $\mathbf{X}^p$ will be very like the original model matrix setup, but make sure you use the predictor variable means, knots, and model matrix column means from the original setup (as stored in `am`) and don't re-compute these things, otherwise you'll get nonsense out.

9. Use your routines to estimate the model

$$\texttt{Volume}_i = \alpha + f_1(\texttt{Girth}_i) + f_2(\texttt{Height}_i) + \epsilon_i$$

   from the data in the `trees` data frame in R. Use `m=1` and `k=10`. Use BFGS in `optim` to estimate the GCV optimal smoothing parameters. Produce plots of the estimated f_1 and f_2 by making use of `am.predict`. If that was too easy, add confidence intervals to the f_j plots, for a bonus mark!