Penalized Natural Cubic Spline Regression

Qingjie Xia

We model f(x) as a **linear combination** of basis functions:

$$f(x) = \sum_{j=1}^{K} \beta_j B_j(x)$$

where:

- $B_j(x)$ are cubic **B-spline basis functions**.
- β_j are the coefficients to estimate.
- K is the number of basis functions (determined by knot selection).

Penalized Regression Formulation

A penalty term is introduced to control overfitting. The objective function is:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int (f''(x))^2 dx$$

- $\sum (y_i - f(x_i))^2$ is the **least squares error**.
- $\lambda \int (f''(x))^2 dx$ penalizes roughness (large second derivatives).
- λ is a **tuning parameter** that controls the trade-off between **fit** and **smoothness**.

- **Design matrix** $\mathbf{B}$ of size $(n \times K)$ with entries $B_j(x_i)$.
- **Second-derivative penalty matrix** $\mathbf{D}$ of size $(K \times K)$, where:

$$D_{jk} = \int B''_j(x) B''_k(x) dx$$

The penalized regression is solved as:

$$(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{D})\beta = \mathbf{B}^T\mathbf{y}$$

Controlling smoothness by penalizing wiggliness

To control the model's smoothness, we could add a 'wiggliness' penalty to the least squares fitting objective.

For example, rather than fitting the model by minimizing

$$\|y - X\beta\|^2 ,$$

it could be fitted by minimizing

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=2}^{k-1} \{f(x^*_{j-1}) - 2f(x^*_j) + f(x^*_{j+1})\}^2,$$

where the summation term measures wiggliness as a sum of squared second differences of the function at the knots (which crudely approximates the integrated squared second derivative penalty used in cubic spline smoothing).

When f is very wiggly the penalty will take high values and when f is 'smooth' the penalty will be low.

If f is a straight line, then the penalty is actually zero.

So the penalty has a null space of functions that are un-penalized: the straight lines in this case.

The dimension of the penalty null space is 2, since the basis for straight lines is 2-dimensional.

The smoothing parameter, λ, controls the trade-off between smoothness of the estimated f and fidelity to the data.

$\lambda \rightarrow \infty$ leads to a straight line estimate for f, while $\lambda = 0$ results in an un-penalized piecewise linear regression estimate.

For the basis of tent functions, it is easy to see that the coefficients of fare simply the function values at the knots, i.e., $\beta_j = f(x_j^*)$.

This makes it particularly straight-forward to express the penalty as a quadratic form, $\beta^T S \beta$, in the basis coefficients (although in fact linearity of f in the basis coefficients is all that is required for this).

Firstly note that

$$\begin{bmatrix} \beta_1 - 2\beta_2 + \beta_3 \\ \beta_2 - 2\beta_3 + \beta_4 \\ \beta_3 - 2\beta_4 + \beta_5 \\ . \\ . \end{bmatrix} = \begin{bmatrix} 1 & -2 & 1 & 0 & . & . & . \\ 0 & 1 & -2 & 1 & 0 & . & . \\ 0 & 0 & 1 & -2 & 1 & 0 & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ . \\ . \end{bmatrix}$$

so that writing the right-hand side as $D\beta$, by definition of $(k-2) \times k$ matrix D, the penalty becomes

$$\sum_{j=2}^{k-1} (\beta_{j-1} - 2\beta_j + \beta_{j+1})^2 = \boldsymbol{\beta}^T \mathbf{D}^T \mathbf{D} \boldsymbol{\beta} = \boldsymbol{\beta}^T \mathbf{S} \boldsymbol{\beta} \qquad (4.5)$$

where $S = D^T D$ (S is obviously rank deficient by the dimension of the penalty null space).

Hence the penalized regression fitting problem is to minimize

$$\|\mathbf{y}-\mathbf{X}\beta\|2 + \lambda\beta^{\mathrm{T}}\mathbf{S}\beta \quad (4.6)$$

w.r.t. β.

The problem of estimating the degree of smoothness for the model is now the problem of estimating the smoothing parameter λ.

But before addressing λ estimation, consider β estimation given λ.

It is fairly straightforward to show that the formal expression for the minimizer of (4.6), the penalized least squares estimator of β, is

$$\hat{\beta} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \boldsymbol{\lambda}\mathbf{S})^{-1}\,\mathbf{X}^{\mathrm{T}}\mathbf{y}. \quad (4.7)$$

Additive models

Now suppose that two explanatory variables, x and v, are available for a response variable, y, and that a simple additive model structure,

$$y_i = \alpha + f_1(x_i) + f_2(v_i) + \varepsilon_i, \quad (4.8)$$

is appropriate. α is an intercept parameter, the f_j are smooth functions, and the ε_i are independent $N(0,\sigma^2)$ random variables.

There are two points to note about this model.

Firstly, the assumption of additive effects is a fairly strong one: $f_1(x) + f_2(v)$ is a quite restrictive special case of the general smooth function of two variables $f(x,v)$.

Secondly, the fact that the model now contains more than one function introduces an identifiability problem: f_1 and f_2 are each only estimable to within an additive constant.

To see this, note that any constant could be simultaneously added to f_1 and subtracted from f_2, without changing the model predictions.

Hence identifiability constraints have to be imposed on the model before fitting.

Provided that the identifiability issue is addressed, the additive model can be represented using penalized regression splines, estimated by penalized least squares and the degree of smoothing selected by cross validation or (RE)ML, in the same way as for the simple univariate model.

1. *Penalized piecewise regression representation of an additive model*

Each smooth function in (4.8) can be represented using a penalized piecewise linear basis. Specifically, let

$$f_1(x) = \sum_{j=1}^{k_1} b_j(x)\delta_j$$

where the δ_j are unknown coefficients, while the $b_j(x)$ are basis functions of the

form (4.4), defined using a sequence of k_1 knots, x^*_j, evenly spaced over the range of x. Similarly

$$f_2(v) = \sum_{j=1}^{k_2} \mathcal{B}_j(v)\gamma_j$$

where the γ_j are the unknown coefficients and the $B_j(v)$ are basis functions of the form (4.4), defined using a sequence of k_2 knots, v^*_j, evenly spaced over the range of v.

Defining n-vector $\boldsymbol{f_1} = [f_1(x_1), \ldots, f_l(x_n)]$, we have $\boldsymbol{f_1} = \boldsymbol{X_1\delta}$ where $b_j(x_i)$ is element i, j of X_1.

Similarly, $\boldsymbol{f_2} = \boldsymbol{X_2\gamma}$, where $B_j(v_i)$ is element i, j of $\boldsymbol{X}_2$.

A penalty of the form (4.5)

$$\sum_{j=2}^{k-1}(\beta_{j-1} - 2\beta_j + \beta_{j+1})^2 = \boldsymbol{\beta}^{\mathrm{T}}\mathbf{D}^{\mathrm{T}}\mathbf{D}\boldsymbol{\beta} = \boldsymbol{\beta}^{\mathrm{T}}\mathbf{S}\boldsymbol{\beta} \qquad (4.5)$$

is also associated with each function:

$\boldsymbol{\delta^T D_1^T D_1 \delta = \delta^T \overline{S}_1 \delta}$ for f_1 and $\boldsymbol{\gamma^T D_2^T D_2 \gamma = \gamma^T \overline{S}_2 \gamma}$ for f_2.

Now it is necessary to deal with the identifiability problem.

For estimation purposes, almost any linear constraint that removed the problem could be used, but most choices lead to uselessly wide confidence intervals for the constrained functions.

The best constraints from this viewpoint are sum-to-zero constraints, such as

$$\sum_{i=1}^{n} f_1(x_i) = 0$$

or equivalently $\mathbf{1}^{\mathrm{T}}\boldsymbol{f}_1 = 0$, where $\mathbf{1}$ is an n vector of 1's.

Notice how this constraint still allows f_1 to have exactly the same shape as before constraint, with exactly the same penalty value.

The constraint's only effect is to shift f_1, vertically, so that its mean value is zero. To apply the constraint, note that we require $\mathbf{1}^{\mathrm{T}}\mathbf{X}_1\boldsymbol{\delta} = 0$ for all $\boldsymbol{\delta}$, which implies that $\mathbf{1}^{\mathrm{T}}\mathbf{X}_1 = 0$.

To achieve this latter condition the column mean can be subtracted from each column of $\mathbf{X}_1$. That is, we define a column centred matrix

$$\widetilde{\boldsymbol{X}}_1 = \boldsymbol{X}_1 - \mathbf{1}\mathbf{1}^T\boldsymbol{X}_1/n$$

and set $\tilde{\boldsymbol{f}}_1 = \widetilde{\boldsymbol{X}}_1\boldsymbol{\delta}$.

It's easy to check that this constraint imposes no more than a shift in the level of $\boldsymbol{f}_1$:

$$\tilde{\boldsymbol{f}}_1 = \tilde{\boldsymbol{X}}_1\boldsymbol{\delta} = \boldsymbol{X}_1\boldsymbol{\delta} - \frac{\mathbf{1}\mathbf{1}^T\boldsymbol{X}_1\boldsymbol{\delta}}{n} = \boldsymbol{X}_1\boldsymbol{\delta} - \mathbf{1}\mathrm{c} = \boldsymbol{f}_1 - c$$

by definition of the scalar $\mathrm{c} = \frac{\mathbf{1}^T\boldsymbol{X}_1\boldsymbol{\delta}}{n}$.

Finally note that the column centring reduces the rank of $\tilde{\boldsymbol{X}}_1$ to $\mathrm{k}_1 - 1$, so that only $\mathrm{k}_1 - 1$ elements of the k_1 vector $\boldsymbol{\delta}$ can be uniquely estimated.

A simple identifiability constraint deals with this problem: a single element of $\boldsymbol{\delta}$ is set to zero, and the corresponding column of $\tilde{\boldsymbol{X}}_1$ and $\mathbf{D}$ is deleted.

The column centred rank reduced basis will automatically satisfy the identifiability constraint.

In what follows the tildes will be dropped, and it is assumed that the $\mathbf{X}_j$, $\mathbf{D}_j$, etc. are the constrained versions.

$$y_i = \alpha + f_1(x_i) + f_2(v_i) + \varepsilon_i, \quad (4.8)$$

Having set up constrained bases for the f_j it is now straightforward to re-express (4.8) as

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

where $\mathbf{X} = (\mathbf{1}, \mathbf{X}_1, \mathbf{X}_2)$ and $\boldsymbol{\beta}^\mathsf{T} = (\alpha, \boldsymbol{\delta}^\mathsf{T}, \boldsymbol{\gamma}^\mathsf{T})$. Largely for later notational convenience it is useful to express the penalties as quadratic forms in the full coefficient vector $\boldsymbol{\beta}$, which is easily done by simply padding out $\bar{\mathbf{S}}_j$ with zeroes, as appropriate. For example,

$$\boldsymbol{\beta}^\mathsf{T}\mathbf{S}_1\boldsymbol{\beta} = (\alpha, \boldsymbol{\delta}^\mathsf{T}, \boldsymbol{\gamma}^\mathsf{T}) \begin{bmatrix} 0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{S}}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \alpha \\ \boldsymbol{\delta} \\ \boldsymbol{\gamma} \end{bmatrix} = \boldsymbol{\delta}^\mathsf{T}\bar{\mathbf{S}}_1\boldsymbol{\delta}.$$

4.3.2 *Fitting additive models by penalized least squares*

The coefficient estimates $\hat{\boldsymbol{\beta}}$ of the model (4.8) are obtained by minimization of the penalized least squares objective

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1 \boldsymbol{\beta}^\mathsf{T} \mathbf{S}_1 \boldsymbol{\beta} + \lambda_2 \boldsymbol{\beta}^\mathsf{T} \mathbf{S}_2 \boldsymbol{\beta},$$

where the smoothing parameters λ_1 and λ_2 control the weight to be given to the objective of making f_1 and f_2 smooth, relative to the objective of closely fitting the response data. For the moment, assume that these smoothing parameters are given.

Similarly to the single smooth case we have

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda_1 \mathbf{S}_1 + \lambda_2 \mathbf{S}_2\right)^{-1} \mathbf{X}^\mathsf{T}\mathbf{y}$$

Test: Choosing the Smoothing Parameter

Note that ten basis functions are used in the fit, including two basis functions for the linear component.

Hence, the fit uses K= 8 spline basis functions.

The combination of low p-value, edf being close to k' and k-index less than 1 is a cause for concern and indicates that a higher number of spline basis functions is needed.

We see that K= 50 passes the k-index test with flying colors.

Finally, a check that K= 25 is also sufficient. Shows a similar k-index score and p-value.

How to check model adequacy?

Checking the model adequacy for Generalized Additive Models (GAMs) fitted using the mgcv package in R is a crucial step to ensure the model is reliable and provides meaningful insights.

Here's a comprehensive guide on how to do it:

1. Visual Inspection of Residuals:

•**plot(model):** The default plot() method for gam objects provides plots of the partial residuals against each predictor. These plots can help assess whether the functional form specified for each predictor (e.g., the smoothness of the spline) is appropriate. Look for patterns in the residuals that might suggest non-linearity not captured by the model or heteroscedasticity (unequal variance).

•**gam.check(model):** This function from the mgcv package produces four diagnostic plots by default:

- **Residuals vs. Fitted Values:** Checks for non-linearity and heteroscedasticity.
- **QQ-plot of Residuals:** Assesses whether the residuals follow the assumed distribution of the response variable.
- **Histogram of Residuals:** Provides another way to check the distribution of residuals.
- **Residuals vs. Linear Predictor:** Similar to residuals vs. fitted values.

•**plot.gam(model, residuals = TRUE, pch = 16, cex = 0.8):** You can also use the plot.gam function with the residuals = TRUE argument to plot the residuals against each predictor.

•**mgcViz package:** The mgcViz package offers enhanced visualization tools for GAMs, including more informative residual plots and ways to explore model fit.

2. Formal Tests:

- **gam.check(model):** In addition to the plots, gam.check() also provides a k-index and a p-value for each smooth term. This test helps assess if the basis dimension (k) chosen for the smooth is sufficient. A low p-value (e.g., < 0.05) might suggest that k is too low and the smooth is too constrained.

- **appraise() function from the gratia package:** The gratia package provides the appraise() function, which offers a more comprehensive set of diagnostic plots and tests for GAMs, including worm plots and quantile-quantile plots with simulation-based confidence intervals.

3. Checking for Autocorrelation:

•If your data has a temporal or spatial structure, you should check for autocorrelation in the residuals. You can use functions like acf() (autocorrelation function) and pacf() (partial autocorrelation function) on the residuals to visualize autocorrelation. If present, you might need to incorporate autocorrelation structures into your model using methods available in mgcv.

4. Assessing Concurvity:

•Concurvity is analogous to multicollinearity in linear models but applies to the smooth terms in GAMs. It occurs when smooth terms are highly correlated, making it difficult to disentangle their individual effects. The concurvity() function in mgcv can be used to assess concurvity in your model. High concurvity can lead to unstable estimates and inflated standard errors.

5. Model Selection and Comparison:

- **Information Criteria (AIC, BIC):** You can use AIC or BIC to compare different GAM models fitted to the same data. Lower values generally indicate a better model fit, but these criteria should be used cautiously with GAMs, especially when comparing models with different smooth terms.

- **Cross-validation:** Cross-validation techniques can provide a more robust assessment of model performance, particularly in terms of predictive accuracy.

Key Considerations:

- **Response Distribution:** Ensure the assumed distribution for your response variable (specified in the family argument of gam()) is appropriate. You can use diagnostic plots and tests to check this.
- **Basis Dimension (k):** The choice of k for each smooth term influences the flexibility of the smooth. If k is too low, the smooth might be too rigid and miss important non-linear patterns. If k is too high, the model might overfit the data. gam.check() can help guide the choice of k.
- **Model Complexity:** GAMs can be flexible, but it's important to avoid overfitting. Regularization methods used in mgcv help with this, but it's still crucial to assess model complexity and ensure it's justified by the data.