Elements of Statistical Learning

Chapter 5
Basis Expansions and Regularization

Review

5.1 INTRODUCTION

Basis Expansions - Idea

Denote by $h_m(X) : \mathbb{R}^p \mapsto \mathbb{R}$ the mth transformation of X, $m = 1, \ldots, M$. We then model

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X), \tag{5.1}$$

For example:

$$h_m(X) = X_m$$

$h_m(X) = X_j^2$ or $h_m(X) = X_j X_k$

$$h_m(X) = log(X_j), \sqrt{X_j}, \cdots$$

$$h_m(X) = I(L_m \leq X_k < U_m)$$

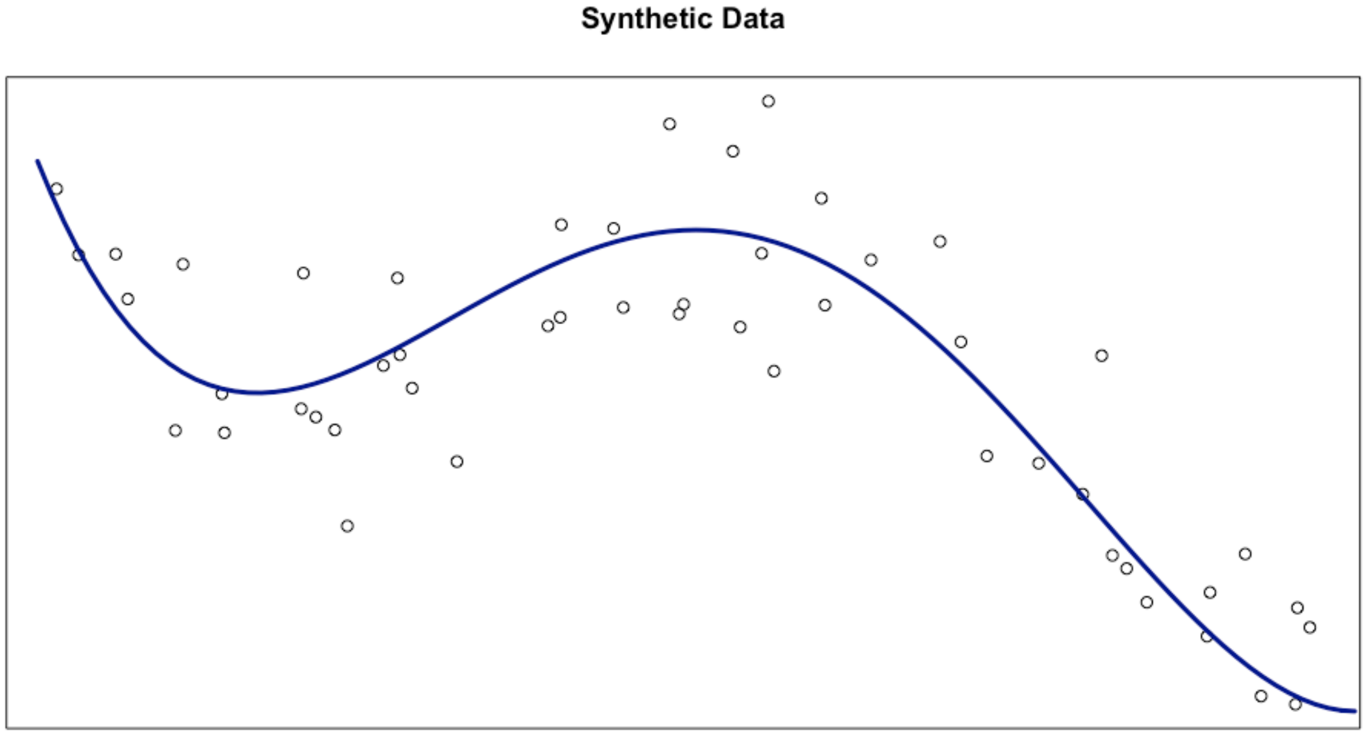
5.2 PIECEWISE POLYNOMIALS AND SPLINES

Consider the case of generating synthetic data from the from the following distribution:

$$X \sim U(0,3)$$

$$Y \sim \frac{1}{4}X^4 - \frac{5}{3}X^3 - \frac{27}{8}X^2 - \frac{9}{4}X + \epsilon$$

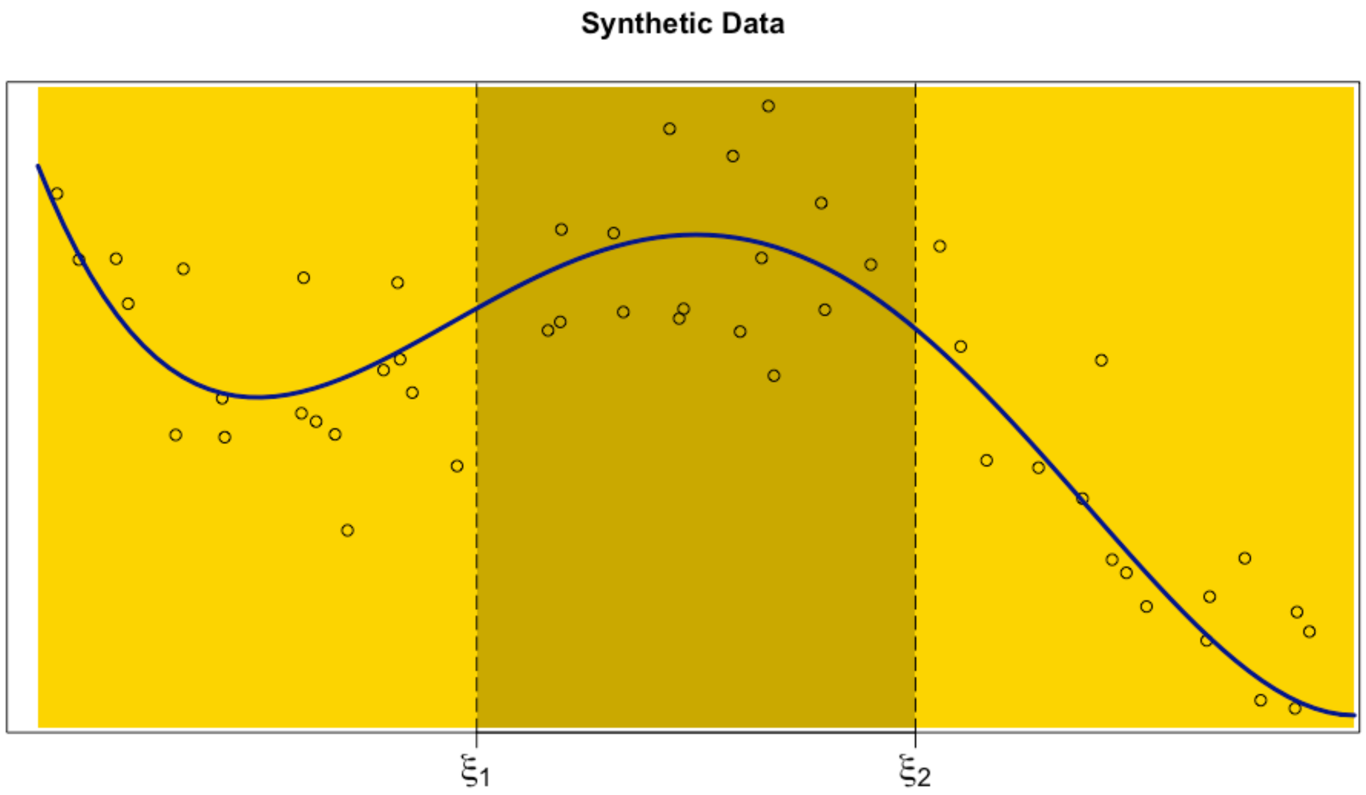
$$\epsilon \sim N(0,0.15)$$



5.2 PIECEWISE POLYNOMIALS AND SPLINES

Now divide X into continuous intervals using indicator basis functions:

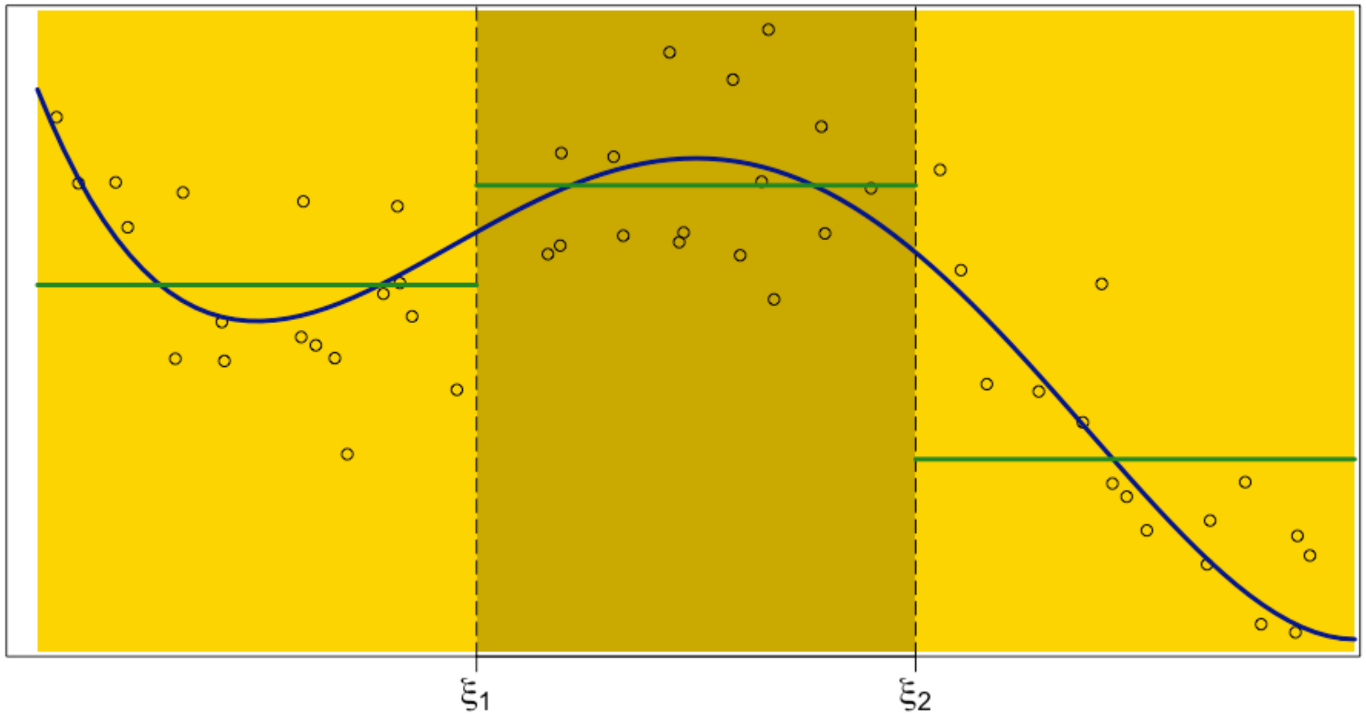
$$h_1(X) = I(X < \xi_1), \quad h_2(X) = I(\xi_1 \leq X < \xi_2), \quad h_3(X) = I(\xi_2 \leq X).$$



5.2 PIECEWISE POLYNOMIALS AND SPLINES

We could fit a piecewise constant to this data (e.g. the mean of each region)

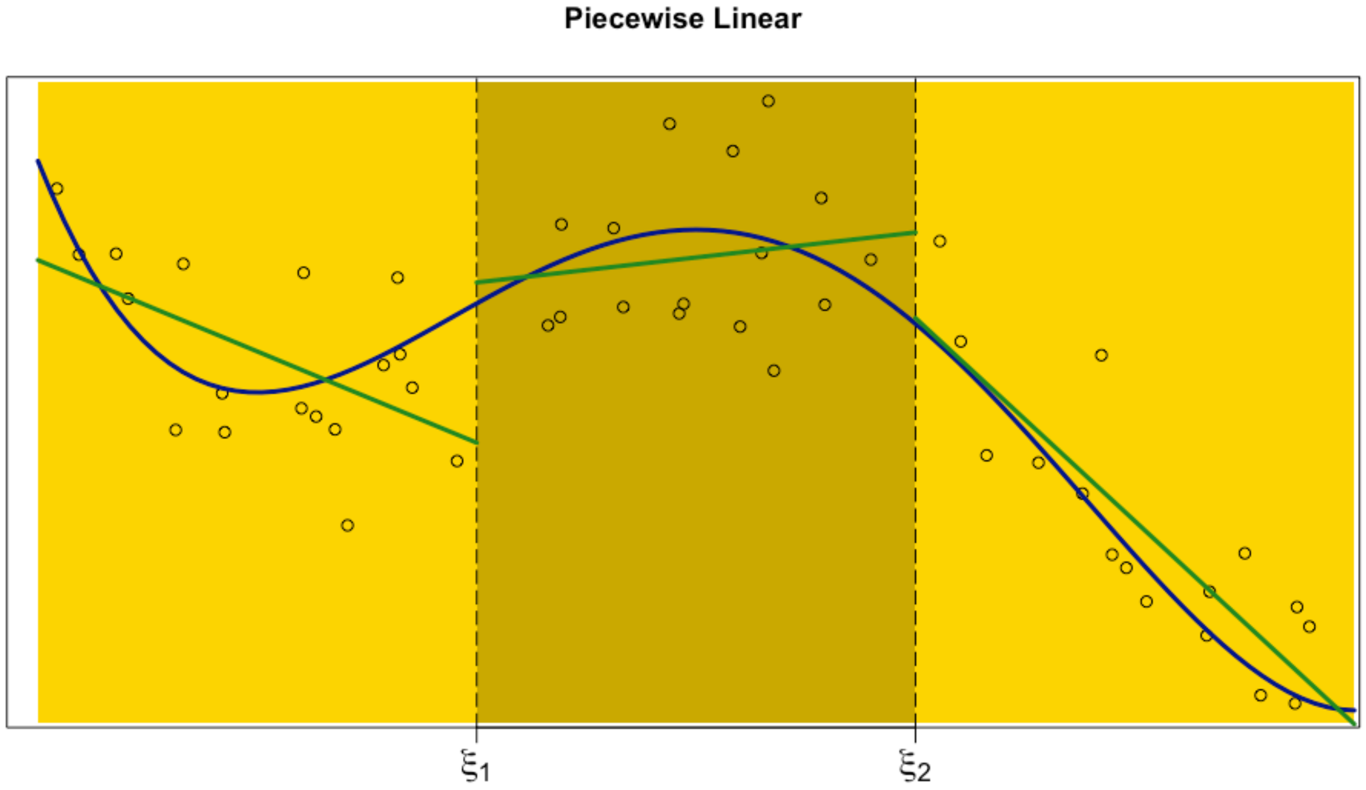
Piecewise Constant



5.2 PIECEWISE POLYNOMIALS AND SPLINES

Even better we could fit a linear model in each region. In this case there would be 6 parameters:

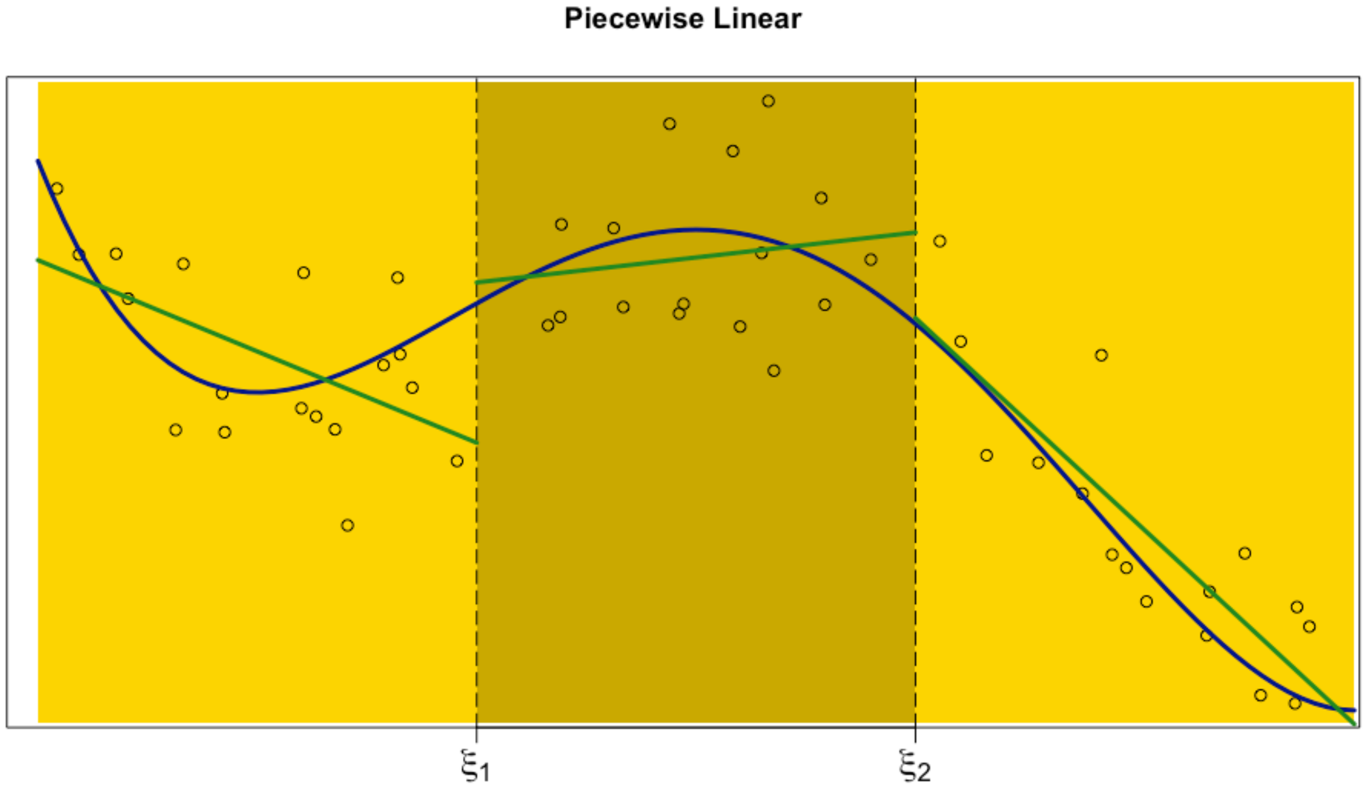
$$\beta_1 I(X < \xi_1) + \beta_2 I(X < \xi_1)X + \beta_3 I(\xi_1 \leq X < \xi_2) + \beta_4 I(\xi_1 \leq X < \xi_2)X + \beta_5 I(\xi_2 \leq X) + \beta_6 I(\xi_2 \leq X)X$$



5.2 PIECEWISE POLYNOMIALS AND SPLINES

We would prefer this to be continuous at the knots e.g.

$$f(\xi_1^-) = f(\xi_1^+) \text{ and } f(\xi_2^-) = f(\xi_2^+)$$



5.2 PIECEWISE POLYNOMIALS AND SPLINES

Lets derive what this might look like

We wish to show that

$$\beta_1 I(X < \xi_1) + \beta_2 I(X < \xi_1)X + \beta_3 I(\xi_1 \leq X < \xi_2) + \beta_4 I(\xi_1 \leq X < \xi_2)X + \beta_5 I(\xi_2 \leq X) + \beta_6 I(\xi_2 \leq X)X$$

with constraints

$$\beta_1 + \xi_1\beta_2 = \beta_3 + \xi_1\beta_4 \qquad (1)$$

$$\beta_3 + \xi_2\beta_4 = \beta_5 + \xi_2\beta_6 \qquad (2)$$

Is equivalent to the following (unconstrained) expression:

$$\alpha_1 + \alpha_2 X + \alpha_3(X - \xi_1)_+ + \alpha_4(X - \xi_2)_+ \qquad (\star)$$

where t_+ denotes the positive part

5.2 PIECEWISE POLYNOMIALS AND SPLINES

We can divide ($\star$) into 3 cases:

$$(a) \qquad X < \xi_1$$
$$\Longrightarrow \alpha_1 + \alpha_2 X \quad = \quad \beta_1 + \beta_2 X$$

$$(b) \qquad \xi_1 \leq X < \xi_2$$
$$\Longrightarrow \alpha_1 - \alpha_3\xi_1 + (\alpha_2 + \alpha_3)X \quad = \quad \beta_3 + \beta_4 X$$

$$(c) \qquad \xi_2 \leq X$$
$$\Longrightarrow \alpha_1 - \alpha_3\xi_1 - \alpha_4\xi_2 + (\alpha_2 + \alpha_3 + \alpha_4)X \quad = \quad \beta_5 + \beta_6 X$$

Now, we just need to equate the beta's with the alpha's…

Setting $\alpha_1 = \beta_1$, $\alpha_2 = \beta_2$, $\alpha_1 - \alpha_3\xi_1 = \beta_3$ and $\alpha_2 + \alpha_3 = \beta_4$ we find that:

$$\beta_1 + \xi_1\beta_2$$
$$= \alpha_1 + \xi_1\alpha_2$$
$$= \alpha_1 - \xi_1\alpha_3 + \xi_1\alpha_3 + \xi_1\alpha_2$$
$$= \beta_3 + \xi_1\beta_4$$

Satisfying constraint (1)

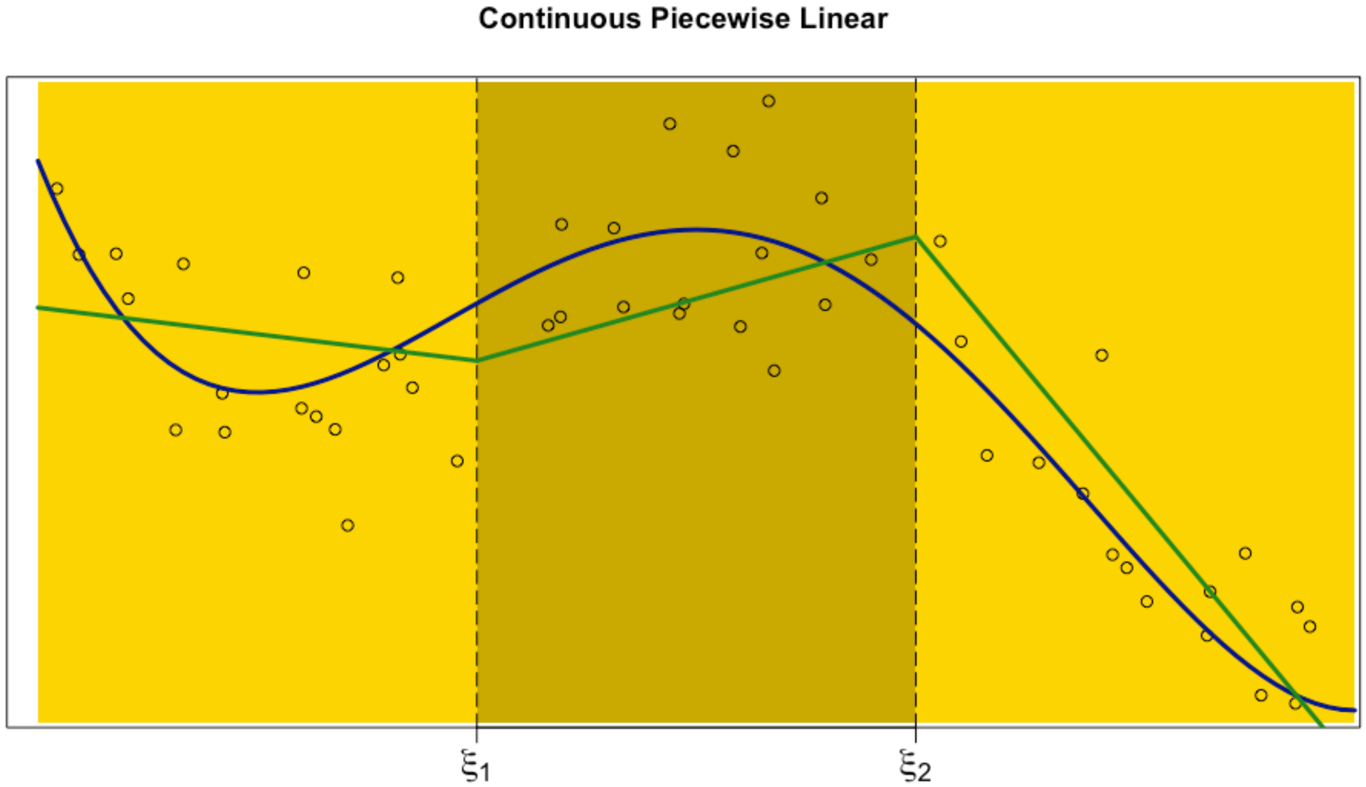
Setting $\alpha_1 - \xi_1\alpha_3 - \xi_2\alpha_4 = \beta_5$ and $\alpha_2 + \alpha_3 + \alpha_4 = \beta_6$ we find that:

$$\beta_3 + \xi_2\beta_4$$
$$= \alpha_1 - \xi_1\alpha_3 + \xi_2(\alpha_2 + \alpha_3)$$
$$= \alpha_1 - \xi_1\alpha_3 - \xi_2\alpha_4 + \xi_2\alpha_4 + \xi_2(\alpha_2 + \alpha_3)$$
$$= \beta_5 + \xi_2\beta_6$$

Satisfying constraint (2)

5.2 PIECEWISE POLYNOMIALS AND SPLINES

And fitting the expression we have derived $(\alpha_1 + \alpha_2 X + \alpha_3 (X - \xi_1)_+ + \alpha_4 (X - \xi_2)_+)$
to the same data we obtain the following fit:



5.2 PIECEWISE POLYNOMIALS AND SPLINES

- We don't have to stop at linear fits, smoother fits can be achieved by increasing the order of the local polynomial.

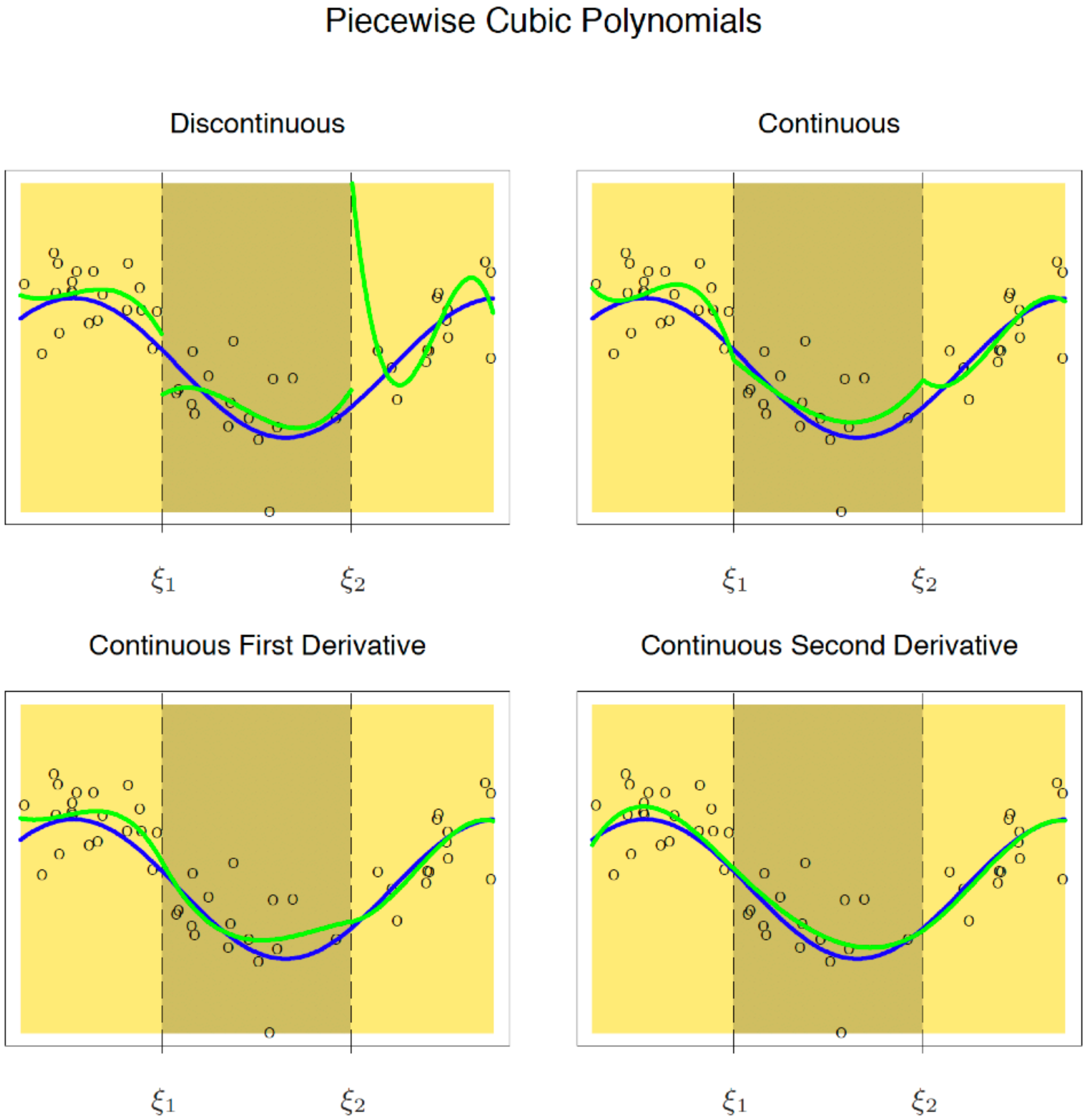
- Fitting a cubic polynomial in each region we can again constrain the function to be continuous ($f(\xi_k^-) = f(\xi_k^+)$).

- Additionally, for a smoother fit we might constrain the first and second derivative to also be continuous at the knots ($f'(\xi_k^-) = f'(\xi_k^+)$ and $f''(\xi_k^-) = f''(\xi_k^+)$).

- A similar derivation to the linear case finds the following *truncated power basis*: $h_1(X) = 1,\ h_2(X) = X,\ h_3(X) = X^2,\ h_4(X) = X^3,\ h_5(X) = (X - \xi_1)_+^3$ and $h_6(X) = (X - \xi_2)_+^3$

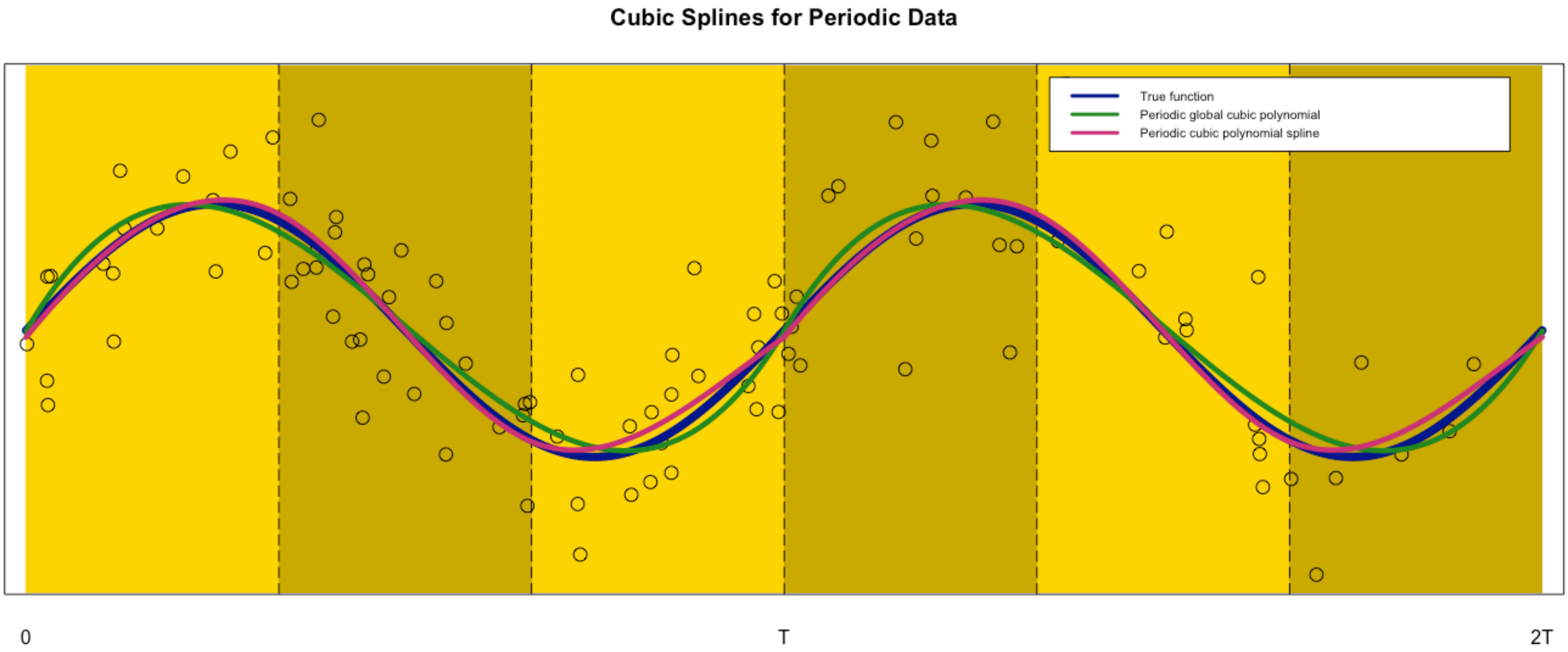
- Parameter count: (3 regions) x (4 parameters per region) - (2 knots) x (3 constraints per knot) = 6

5.2 PIECEWISE POLYNOMIALS AND SPLINES



5.2 PIECEWISE POLYNOMIALS AND SPLINES

Example - Periodic Data



5.2 PIECEWISE POLYNOMIALS AND SPLINES

Natural Cubic Splines

A general form for the *truncated power basis of order* M with K knots turns out to be:

$$h_j(X) = X^{j-1}, \quad j = 1, \cdots, M$$

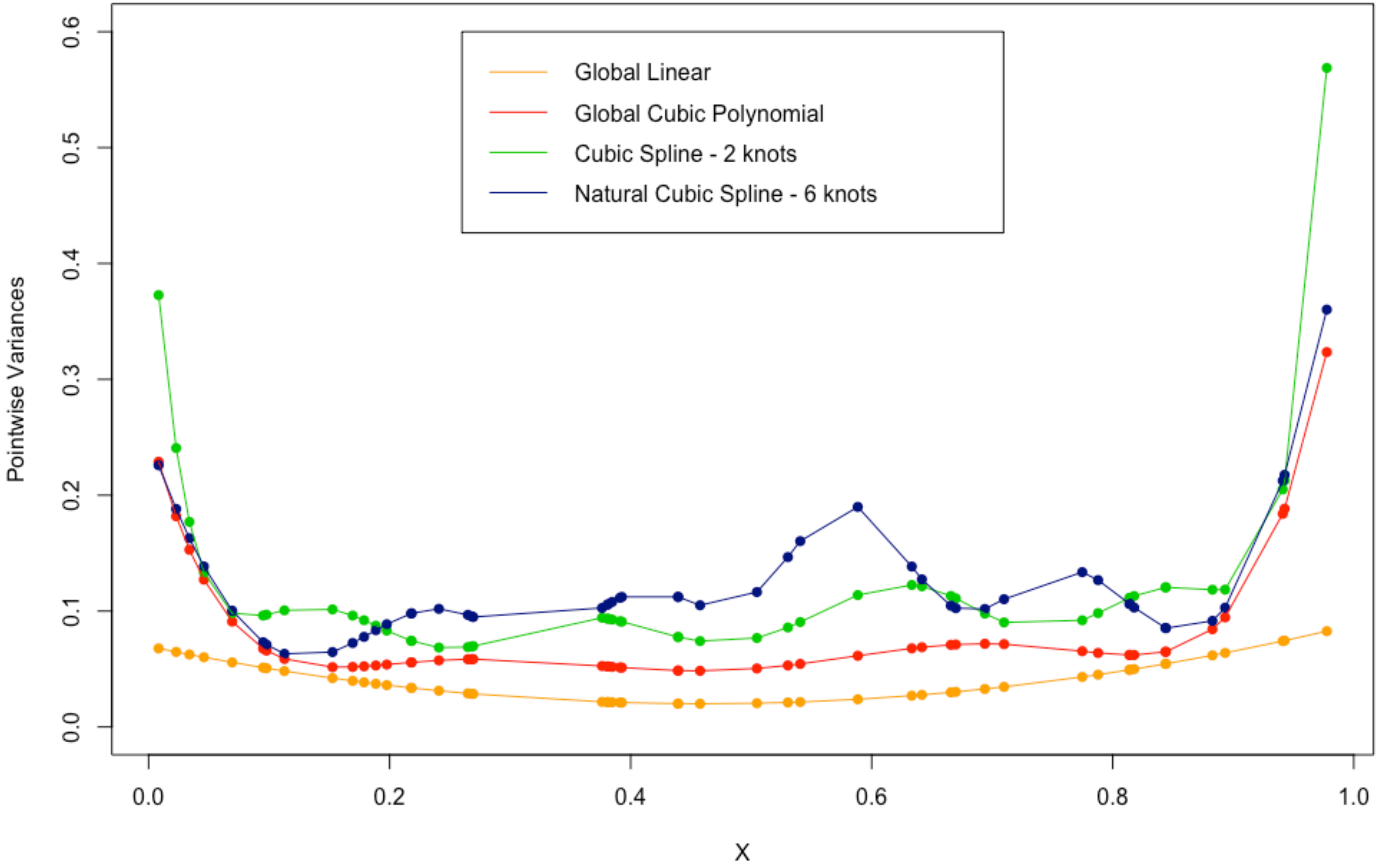
$$h_{M+l}(X) = (X - \xi_l)_+^{M-1}, \quad l = 1, \cdots, K$$

<u>Drawback</u>: Polynomials tend to be erratic (high variance) near the boundaries. The problem is exacerbated with splines.

<u>Solution</u>: Constrain the function to be *linear beyond the boundary knots*.

5.2 PIECEWISE POLYNOMIALS AND SPLINES

Natural Cubic Splines



$$X \sim U(0,1)$$

$$Y \sim X + \epsilon$$

$$\epsilon \sim N(0,1)$$

5.2 PIECEWISE POLYNOMIALS AND SPLINES

Natural Cubic Splines

A natural cubic spline with K knots is represented by K basis functions.

For example, starting from the truncated power series basis and imposing the boundary constraints of linearity beyond the boundary knots we arrive at:

$$N_1(X) = 1, \quad N_2(X) = X, \quad N_{k+2}(X) = d_k(X) - d_{K-1}(X)$$

where:

$$d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k}$$

5.4 SMOOTHING SPLINES

Consider the following problem: among all functions $f(x)$ with two continuous derivatives, find one that minimises the penalised residual sum of squares

$$RSS(f, \lambda) = \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt$$

Remarkably, it can be shown that this has an explicit, finite-dimensional, unique minimiser which is a **natural cubic spline** with knots at the unique values of the

x_i, $i = 1,2,\cdots,N$

5.4 SMOOTHING SPLINES

Sketch proof

Consider the data $a < x_1 < \cdots < x_N < b$ with $N \geq 2$. Suppose that g is a natural cubic spline with knots at every x_i , let $\tilde{g}$ be any other differentiable function on $[a, b]$ and define

$$h(x) = \tilde{g}(x) - g(x).$$

3 Parts

(a) We can use integration by parts and the fact that g is a cubic spline to prove that:

$$\int_a^b g''(x)h''(x)dx = 0$$

(b) Using this fact we can show that:

$$\int_a^b \tilde{g}''(t)^2 dt = \int_a^b [h''(t) + g''(t)]^2 dt = \int_a^b h''(t)^2 dt + \cancel{2\int_a^b g''(t)h''(t)dt} + \int_a^b g''(t)^2 dt \geq \int_a^b g''(t)^2 dt$$

5.4 SMOOTHING SPLINES

Sketch proof

Consider the data $a < x_1 < \cdots < x_N < b$ with $N \geq 2$. Suppose that g is a natural cubic spline with knots at every x_i , let $\tilde{g}$ be any other differentiable function on $[a, b]$ and define

$$h(x) = \tilde{g}(x) - g(x).$$

3 Parts

(c) Returning to the penalised least squares problem:

$$\min_f \left[\underbrace{\sum_{i=1}^{N} \{y_i - f(x_i)\}^2}_{(1)} + \lambda \underbrace{\int \{f''(t)\}^2 dt}_{(2)} \right]$$

We argue that, by the definition of a natural spline interplant to every data point, (1) is minimised by g and, by part (b), (2) is also minimised by g.

5.4 SMOOTHING SPLINES

Since the solution is a natural spline ($f(x) = \sum_{j=1}^{N} N_j(x)\theta_j$), the penalised RSS criterion can be rewritten as:

$$RSS(\theta, \lambda) = (\mathbf{y} - \mathbf{N}\theta)^T(\mathbf{y} - \mathbf{N}\theta) + \lambda\theta^T\Omega_{\mathbf{N}}\theta$$

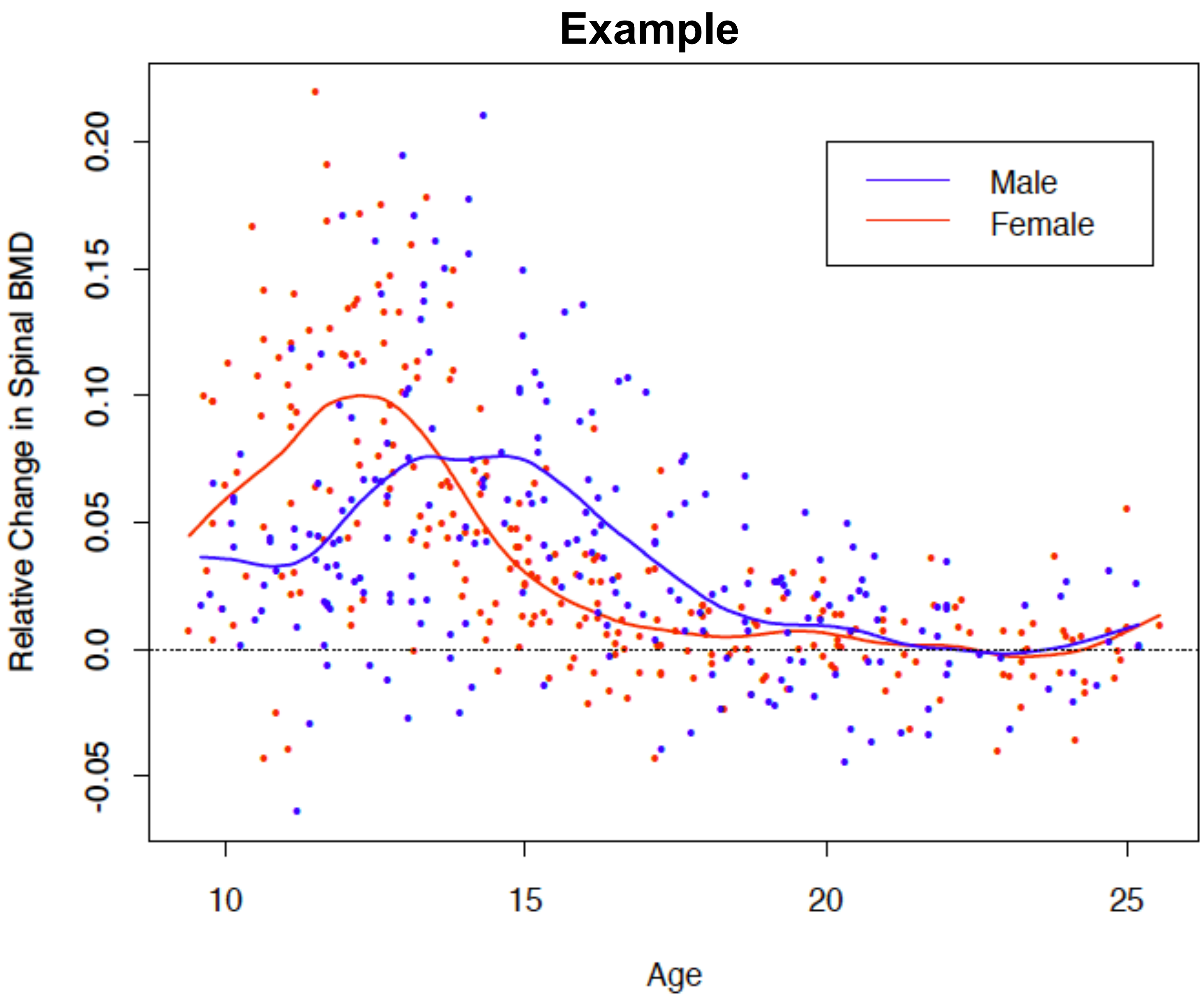
Where $\{\mathbf{N}\}_{ij} = N_j(x_i)$ and $\{\Omega_{\mathbf{N}}\}_{jk} = \int N_j''(t)N_k''(t)dt$, and the solution can easily be seen to be:

$$\hat{\theta} = (\mathbf{N}^T\mathbf{N} + \lambda\Omega_{\mathbf{N}})^{-1}\mathbf{N}^T\mathbf{y}$$

But is this over parameterised?

No! The penalty term translates to a penalty on the spline coefficients, which are shrunk some of the way toward the linear fit.

5.4 SMOOTHING SPLINES



5.4 SMOOTHING SPLINES

A smoothing spline with a prechosen λ is an example of a **linear smoother**

$$\begin{aligned}\hat{\mathbf{f}} &= \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\Omega_{\mathbf{N}})^{-1}\mathbf{N}^T\mathbf{y} \\ &= \mathbf{S}_\lambda \mathbf{y}\end{aligned}$$

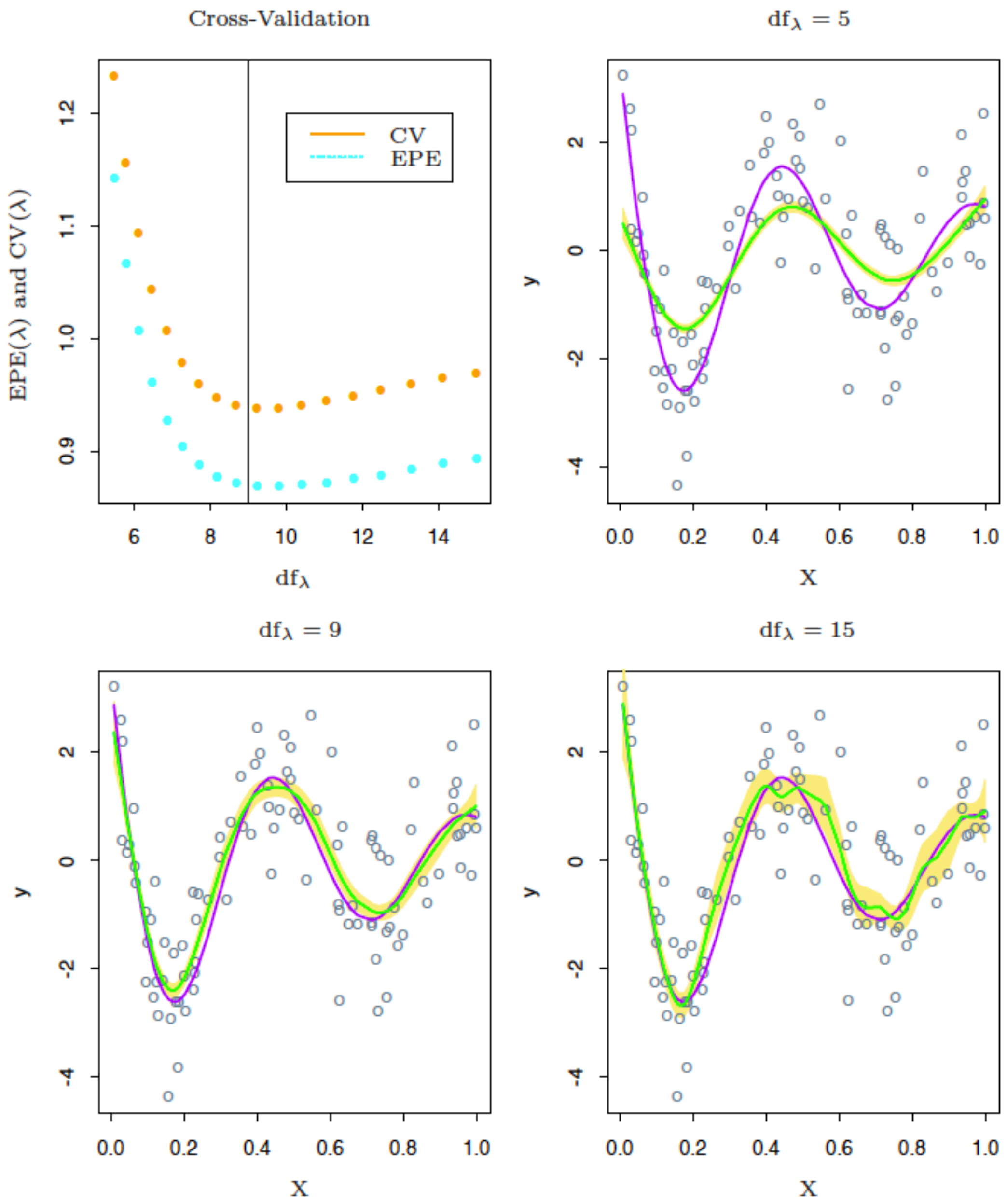
We can think of $\mathbf{S}_\lambda$ as the *smoother matrix* operating on the vector $\mathbf{y}$. This is analogous so the hat matrix ($\mathbf{H}$) in the linear regression setting. Increasing the value of λ corresponds to more smoothing.

In linear regression the degrees of freedom was defined as $trace(\mathbf{H})$. By analogy we define the **effective degrees of freedom** of a smoothing spline to be:

$$df_\lambda = trace(\mathbf{S}_\lambda)$$

Since $df_\lambda = trace(\mathbf{S}_\lambda)$ is monotone in λ for smoothing splines, we can invert the relationship and specify λ by fixing df.

5.5 AUTOMATIC SELECTION OF SMOOTHING PARAMETERS



5.5 AUTOMATIC SELECTION OF SMOOTHING PARAMETERS

The Bias-Variance Tradeoff

The expected prediction error is a natural quantity of interest

$$\begin{aligned} EPE(\hat{f}_\lambda) &= E(Y - \hat{f}_\lambda(X))^2 \\ &= Var(Y) + E[Bias^2(\hat{f}_\lambda(X)) + Var(\hat{f}_\lambda(X))] \\ &= \sigma^2 + MSE(\hat{f}_\lambda) \end{aligned}$$

Unfortunately, we do not have access to EPE, and need an estimate. Leave-one-out cross validation is a common approach and, remarkably, we can calculate its score for a given λ with *a single fit* on the data

$$\begin{aligned} CV(\hat{f}_\lambda) &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}_\lambda^{(-i)}(x_i))^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{f}_\lambda(x_i)}{1 - S_\lambda(i,i)}\right)^2 \end{aligned}$$

5.6 NONPARAMETRIC LOGISTIC REGRESSION

It is straightforward to transfer the smoothing spline approach to other domains. For example, consider the logistic regression model:

$$log \frac{Pr(Y = 1 | X = x)}{Pr(Y = 0 | X = x)} = f(x)$$

Which implies:

$$Pr(Y = 1 | X = x) = \frac{e^{f(x)}}{1 + e^{f(x)}}$$

Just as for regular logistic regression, we can construct the penalised log-likelihood criterion. Again we find the optimal f is a finite-dimensional natural spline with knots at the unique values of x.

We can apply Newton-Raphson to iteratively solve for $\theta^{new} \leftarrow \theta^{old}$ and, similar to what we have seen in the previous slides, we can express this update in terms of fitted values and a smoother matrix $f^{new} = \mathbf{S}_{\lambda,\omega}\mathbf{z}$.

5.7 MULTIDIMENSIONAL SPLINES

The procedures that we have discussed generalise to higher-dimensional x.

For natural splines we have two options for creating our basis:

(a) *Additive natural splines* - Simply fit natural splines in each of the dimensions.

(b) *Tensor product basis* - For example in $\mathbb{R}^2$, suppose we have a set of M_1 basis functions for X_1 and M_2 basis functions for M_2. Then the $M_1 \times M_2$ dimensional tensor product basis is defined by:

$$g_{jk}(X) = h_{1j}(X_1)h_{2k}(X_2), \quad j = 1,\cdots, M_1, \quad k = 1,\cdots, M_2$$

5.7 MULTIDIMENSIONAL SPLINES

Smoothing splines generalise to higher dimensions as well. In $\mathbb{R}^d$, we seek a d-dimensional regression function $f(x)$ solving:

$$\min_f \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda J[f] \qquad (\star)$$

where J is an appropriate penalty functional for stabilising a function f. For example in $\mathbb{R}^2$ a natural generalisation of the one dimensional case is:

$$J[f] = \int\int_{\mathbb{R}^2} [(\frac{\partial^2 f(x)}{\partial x_1^2})^2 + 2(\frac{\partial^2 f(x)}{\partial x_1 \partial x_2}) + (\frac{\partial^2 f(x)}{\partial x_2^2})] dx_1 dx_2$$

Optimising $(\star)$ with this penalty leads to a smooth two dimensional surface known as a **thin-plate spline**.

5.9 WAVELET SMOOTHING

With smoothing splines, we use a complete basis, but then shrink the coefficients toward smoothness. *Wavelets* typically use a complete orthonormal basis to represent functions, but then shrink and select the coefficients toward a sparse representation.

Wavelets bases are very popular in signal processing and compression, since they are able to represent both smooth and/or locally bumpy functions in an efficient way