A Short Course in Machine Learning (Part III)

Reza Ramezan

Department of Statistics and Actuarial Science

University of Waterloo

rramezan@uwaterloo.ca

www.neuroinformatics.ca

Alzahra University

March 4–5, 2021

Agenda

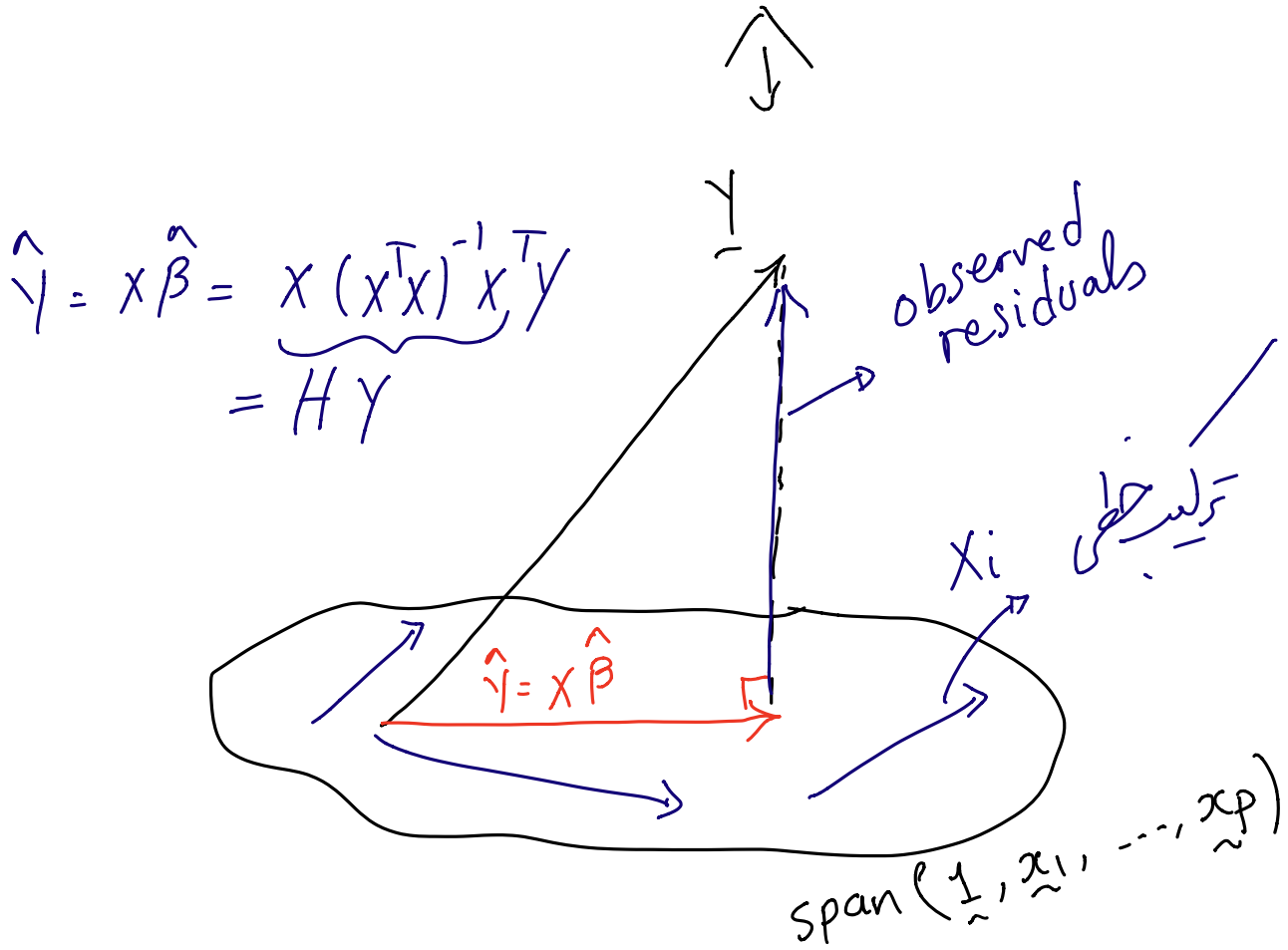
- Introduction to Smoothing Methods
  - Polynomial Regression
  - Basis Expansion
- Splines
  - Cubic, Natural, B, and the underlying geometry/linear algebra
- Regularization Methods
  - Shrinkage Estimators: Ridge, LASSO, Elastic Net
  - Smoothing Splines
- Local Weighting
  - ~~Delta Neighbourhood and KNN~~
  - Kernel Smoothing
  - Local Smoothing and LOESS
- Multidimensional Smoothing (if time permits)
  - Tensor Product Bases
  - Thin-plate Splines
  - Additive Spline Model

Housekeeping Stuff

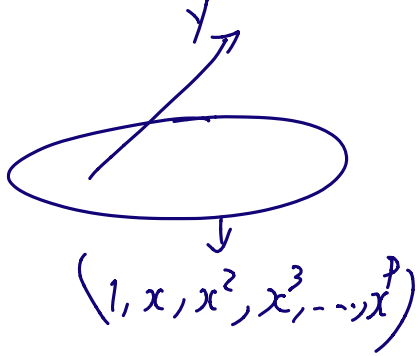
- **Format:** This part of the workshop is mostly applied with minimal theoretical content. More details can be found in the references below.

- **Additional Resources:**
  - An Introduction to Statistical Learning by James *et al.* : Ythe book, lectures by the authors, codes, errata, etc. under this link.
  - The elements of Statistical Learning by Hastie, Tibshirani, and Friedman : the book is available here, and more stuff (codes, data, errata, etc.) can be found here.
  - Statistical Learning with Sparsity by Hastie, Tibshirani, and Wainwright available here.
  - Machine Learning: A Probabilistic Perspective by Kevin Murphy $\rightarrow$ (has a computer science approach)
  - A Practical Guide to Splines by Carl de Boor

Geometry of $Y|(X = x) = \mu(x) + \epsilon$



Revisiting the Geometry of $Y|(X = x) = \mu(x) + \epsilon$



In polynomial regression, the model

$$\mu(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_p x^p$$

is a linear combination of *polynomial* terms of x.

However, $\mu(x)$ can be expressed as some (unknown) linear combination of other functions, e.g.

$$\mu(x) = \theta_1 g_1(x) + \theta_2 g_2(x) + \theta_3 g_3(x) + \theta_4 g_4(x), \quad \theta_i \in \mathbb{R}$$

where $g_1(x) = 1$, $g_2(x) = x$, $g_3(x) = \sin(x)$, and $g_4(x) = \log(x)$.

Revisiting the Geometry of $Y|(X = x) = \mu(x) + \epsilon$

- All linear combinations of these four specific functions $g_1, \cdots, g_4$ form a *subspace*.
- The functions $g_1, \cdots, g_4$ are *generators* of that subspace.
- If the generators are linearly independent of one another (orthogonal) then these functions also form a set of *orthogonal basis functions* for that subspace.
- The *linear model* asserts that $\mu(x)$ lies in that subspace, subspace of dimension equal to the number of basis functions which define it.

An Infinite-Dimensional Basis

$$f(x) = \sum_{i=0}^{\infty} \frac{f^{(i)}(x_0)(x-x_0)^i}{i!}$$

- One famous/easy example of basis functions is the Taylor expansion. For example, recall the Taylor expansion of e^x about the point $x_0 = 0$, i.e.

$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \cdots$$

- Interpretation: exponential function is a known (infinite) linear combination of all the (infinitely many) simple polynomials in x.

- These polynomials form an *infinite-dimensional basis* for the space of real-valued functions of a real-valued variable.

A Model Based on an Infinite-Dimensional Basis

- More approperiately, we should write

$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + R(x)$$

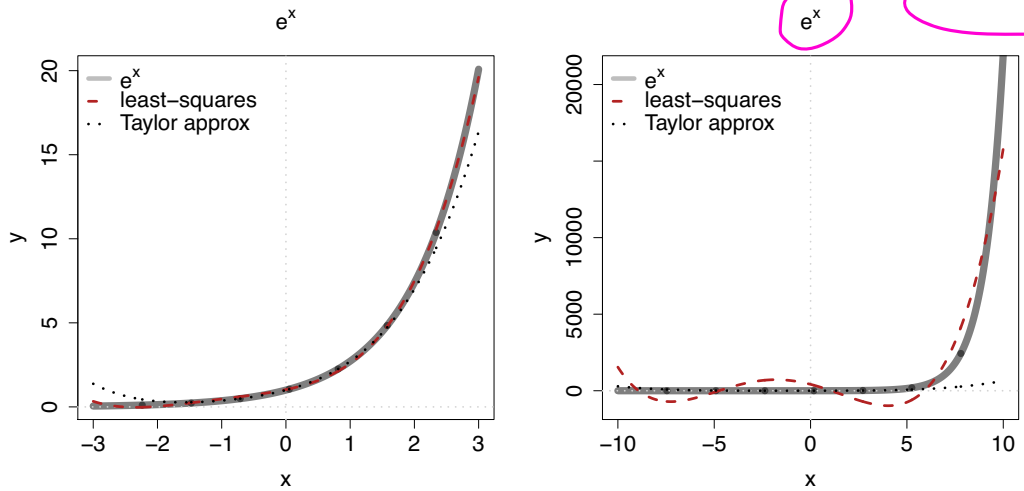
  where $R(x)$ is a $o(x^4)$ function, i.e. $\lim_{x \to 0} \frac{R(x)}{x^4} = 0$.

- Hence, an approximation of e^x is

$$e^x \approx \theta_0 x^0 + \theta_1 x + \cdots + \theta_4 x^4$$

- In the data analytics world, we don't know the form of $\mu(x)$ hence cannot write the Taylor expansion (the constants!), but we can estimate them.

Example : e^x



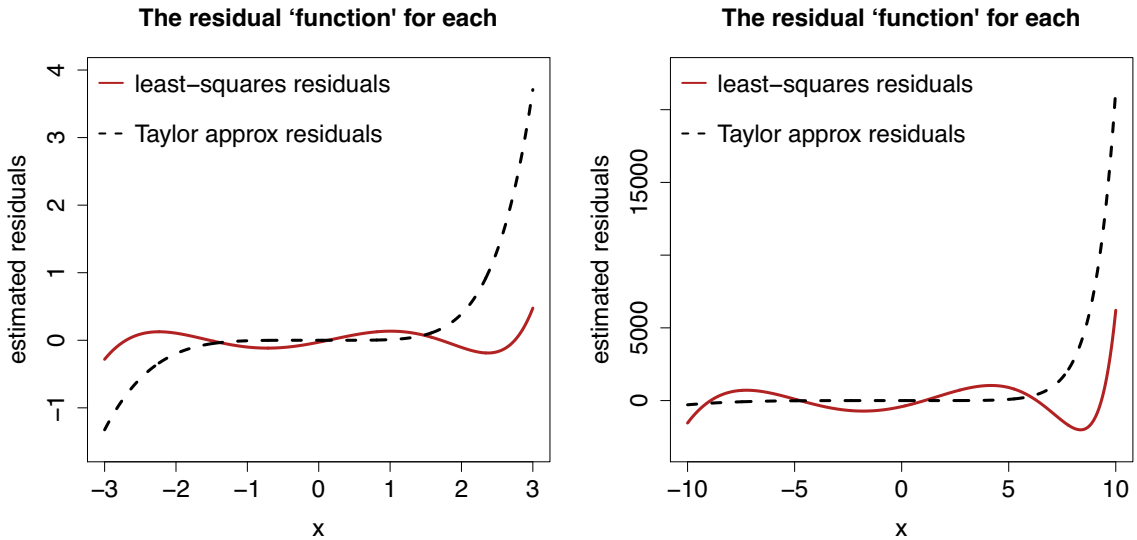
- Note that the least-squares fit performs better than the Taylor series approximation
- However, this Taylor approximation is "tailored" to $x = 0$ (pun intended!).... Taylor approximation outperforms LS fit in $x \in (-1, 1)$ based on the fit $x \in (-2, 2)$.
- Don't get too excited about the polynomial model though (panel on the right)

Example : e^x

$y = f(x) + \varepsilon$

residual $= y - \hat{y} = y - \hat{f}(x)$

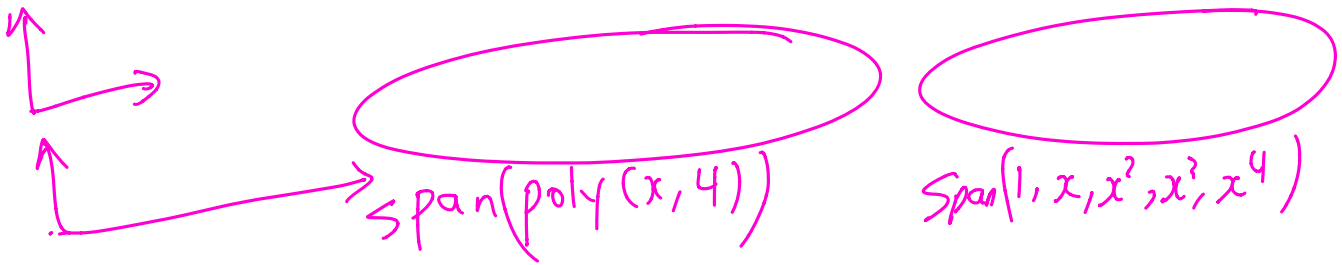
We can also have a look at what is "left over" thefrom our fitted functions, i.e. the "residual function:"



Basis Functions for e^x

$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \varepsilon$ $\quad \hat{\beta} = (X^T X)^{-1} X^T y$

- `lm(y ~ x + I(x^2) + I(x^3) + I(x^4))` uses simple polynomials x^0, x^1, x^2, x^3, and x^4
- `lm(y ~ poly(x,4))` uses *orthogonal polynomials* which generate the same subspace.
- The latter case form an *orthogonal basis* for the subspace.
- $\widehat{\mu}(x)$, i.e. the fitted values are the same based on the two implementations, but the model coefficients are different because **two different sets of basis functions** are used.



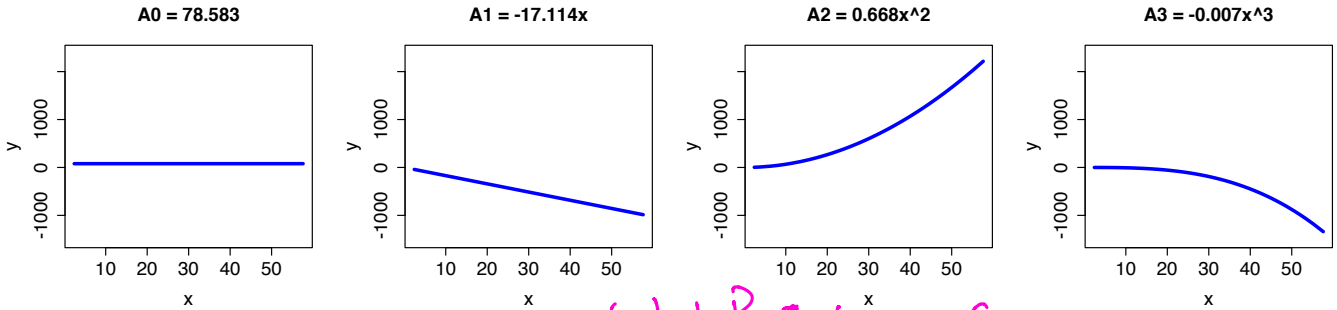
Basis Functions for e^x

```
> x <- seq(-3, 3, length.out = 1000)
> y <- exp(x)

> fit1 <- lm(y~x + I(x^2) + I(x^3) + I(x^4))
> fit2 <- lm(y~poly(x,4))
> Taylor = sapply(0:4, function(x) 1/factorial(x))

> round(fit1$coefficients,3)
(Intercept)           x      I(x^2)      I(x^3)      I(x^4)
      1.029       0.788       0.435       0.269       0.062

> round(fit2$coefficients,3)
(Intercept) poly(x, 4)1 poly(x, 4)2 poly(x, 4)3 poly(x, 4)4
      3.346     123.146      77.778      34.882      12.153

> round(Taylor, 3)
[1] 1.000 1.000 0.500 0.167 0.042
```

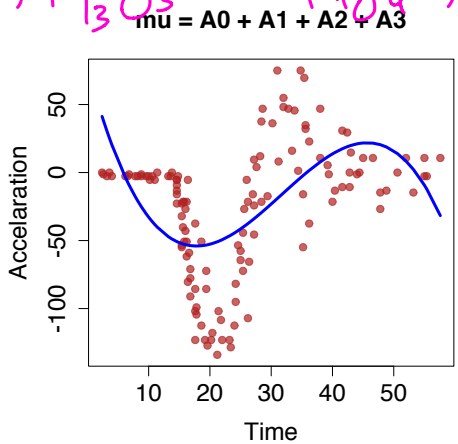
Note: as you add polynomial terms to `fit1`, its coefficients get closer and closer to `Taylor`. Why?

Example

```
> fit <- lm(accel ~ times+I(times^2)+I(times^3) , data=mcycle)
> round(coef(fit),3)
(Intercept)       times  I(times^2)  I(times^3)
     78.583     -17.114       0.668      -0.007
```



$$y = \beta_0 + \beta_1 g_1(x) + \beta_2 g_2(x) + \beta_3 g_3(x) + \beta_4 g_4(x) + \varepsilon$$

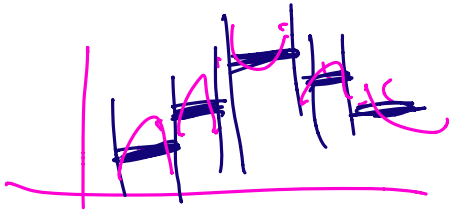


Generalization

$= \beta_0 + \sum_i \beta_i g_i(x)$ $= HY$

- In regression problems, $\mu(X) = E(Y|X)$ will typically be non-linear and non-additive in X.
- We represent $\mu(X)$ by a linear model for convenience, and sometimes out of necessity.
- Linear models are convenient because they are easy to interpret and sometimes necessary because with small n and large p a linear model might be all we could fit the data without over-fitting.

Generalization



Basic Idea: Replace/augment the vector of explanatory variates $\mathbf{X}$ with transformations of $\mathbf{X}$ and use linear models in the new space of explanatory variables. We will use

- **Polynomial regression** add extra predictors in the form of powers of X, i.e. X, X^2, X^3, ...
- **Step functions** cut the range of a variable into K distinct regions, and calculate the mean (or other attributes) of the response function as the fit. This is a piecewise constant fit.

Generalization

- **Regression splines** extension of polynomials and piecewise constant fit, and more flexible. The range of a variable X is divided into K distinct regions. Within each region a polynomial function us fit to the data. However, polynomials are constrained so that they join smoothly at the boundary of the regions a.k.a. *knots*. The flexibility of the fit depends on the number of knots.
- **Smoothing splines** are similar to regression splines, but are resulted from minimizing a residual sum of squares criterion subject to a smoothness penalty.
- **Local regression** is similar to splines, but the regions are allowed to overlap and indeed they do so in a very smooth way.
- **Generalized additive models** extend the methods above to deal with multiple predictors.

Linear Basis Expansion

We mentioned that the basic idea is to replace/augment the vector of explanatory variates $\mathbf{X}$ with transformations of $\mathbf{X}$ and use linear models in the new space of explanatory variables.

Denote by $h_m(\mathbf{X}) : \mathbb{R}^p \to \mathbb{R}$ the m^{th} transformation of $\mathbf{X}$, $m = 1, ..., M$. We then write a *linear basis expansion* in $\mathbf{X}$

$$\mathbf{Y}|\mathbf{X} = \mu(\mathbf{X}) + \epsilon = \sum_{m=1}^{M} \beta_m h_m(\mathbf{X}) + \epsilon$$

generator

Note that once the basis functions h_m have been determined (the "dictionary" of functions), the models are linear in these new variables, and fitting proceeds as before.

Choosing the "Dictionary"

Question: How does one choose the basis functions for the model?

Answer: This is similar to a variable-selection problem. Some options are:

- Choose them manually beforehand to limit the class of functions (very popular)
  - e.g. additive models:
    $\mu(\mathbf{X}) = \sum_{j=1}^{p} \mu_j(X_j) = \sum_{j=1}^{p} \sum_{m=1}^{M_j} \beta_{jm} h_{jm}(X_j)$
- Include all and use a variable-selection procedure
- Include all and regularize (e.g. Ridge, LASSO, Elastic Net) to shrink the coefficients.

Basis Functions

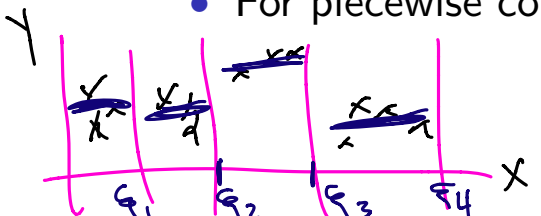
- In a more general case, let's represent the functions h with b (for basis). As discussed, the idea is to have a family of functions or transformations $b_1(X),...,b_K(X)$ to fit a linear model

$$y_i = \beta_0 + \sum_{j=1}^{K} \beta_j b_j(x_i) + \epsilon_i$$

- Note that while the model above is not necessarily linear in X, it is linear in the basis functions b_j, which are **fixed** and **known** functions.

finite set of polynomials

- Examples
  - For polynomial regression: $1, x, x^2, x^3, \ldots$
  - For piecewise constant regression: $b_j(x) = I(\xi_{j-1} \leq x < \xi_j)$



indicator functions

$j = 1, \ldots, k$

Fitting a Model with Basis Functions

- We can use ML or LS (or other approaches) to fit the linear model in the previous slide $\rightarrow$ predictors are .....
- All inference for linear models (t-test, F-test for model's significance, ANOVA, etc.) are valid.
- We have many choices for basis functions, e.g.
  - Wavelets
  - Fourier
  - Splines

Harr Basis

Agenda

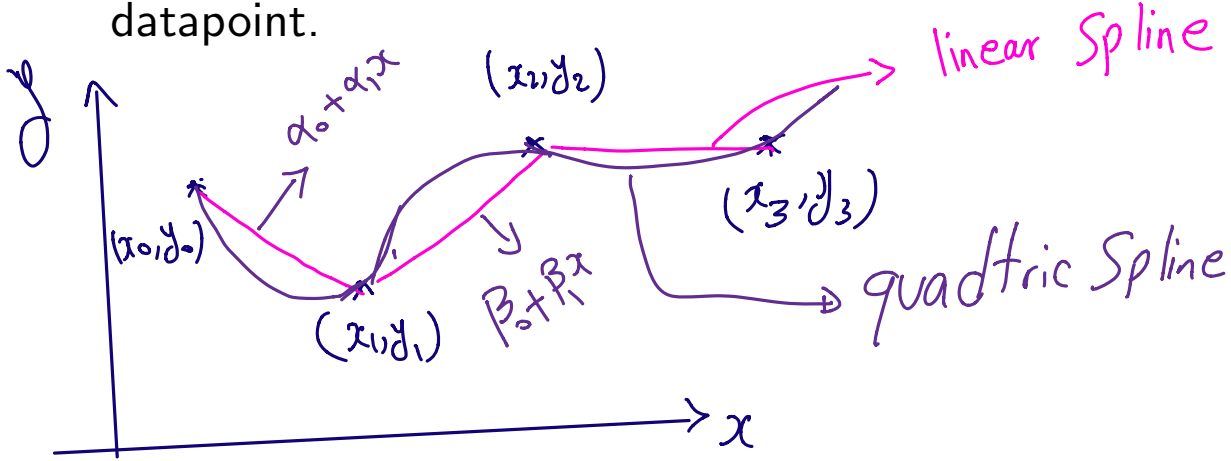
- Introduction to Smoothing Methods
  - Polynomial Regression ✓
  - Basis Expansion ✓
- Splines
  - Cubic, Natural, and B-splines
- Regularization Methods
  - Shrinkage Estimators: Ridge, LASSO, Elastic Net
  - Smoothing Splines
- Local Weighting
  - ~~Delta Neighbourhood and KNN~~
  - Kernel Smoothing
  - Local Smoothing and LOESS
- Multidimensional Smoothing (if time permits)
  - Tensor Product Bases
  - Thin-plate Splines
  - Additive Spline Model

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

What Is a Spline?

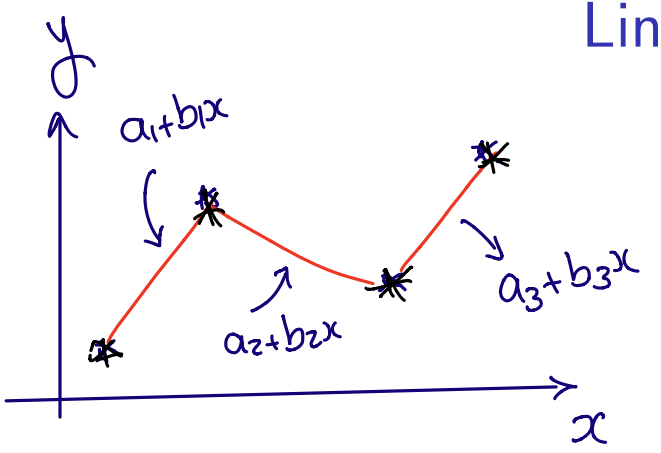
↳ perfect fit

- For a moment forget about statistical fit, and think about a perfect fit, i.e. a model which goes through each and every datapoint.



- A spline is a piecewise function which explains the behaviour of this "perfect fit."

Linear Spline



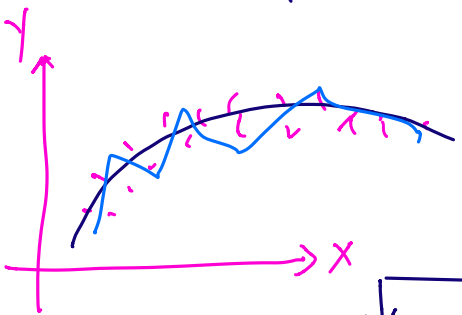
Suppose that the data is

$(x_0, y_0), \ldots, (x_n, y_n)$

↳ $n+1$ points.

$n+1$ points $\Rightarrow$ n lines

each line has 2 parameters $\Rightarrow$ $2n$ parameters



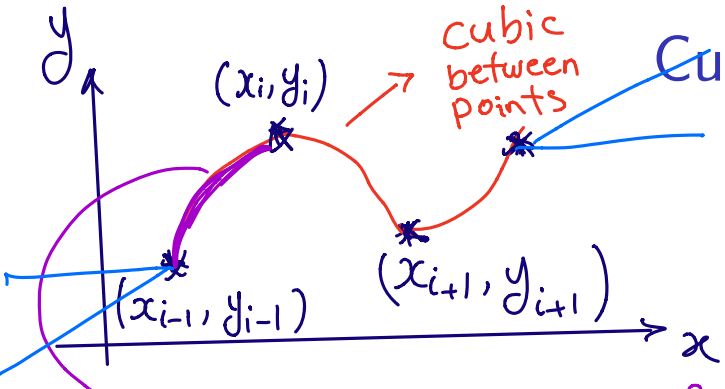
$\Downarrow$

need $2n$ equations

Continuity constraint

$$\begin{cases} a_i + b_i x_i = y_i & i = 1, \ldots, n \\ a_i + b_i x_{i-1} = y_{i-1} & i = 1, \ldots, n \end{cases} \Rightarrow 2n \text{ equations}$$

Cubic Spline



$(x_0, y_0), \ldots, (x_n, y_n) \rightarrow$ $n+1$ data points

$\Rightarrow$ n cubic curves (each with 4 pram.)

$\Rightarrow$ $4n$ parameters

$\Rightarrow$ need $4n$ equations

Constraints:

Continuity
$$\begin{cases} S_i(x_i) = y_i & i = 1, \ldots, n \rightarrow n \text{ equations} \\ S_i(x_{i-1}) = y_{i-1} & i = 1, \ldots, n \rightarrow n \text{ equations} \end{cases}$$

Smoothness
$$S_i'(x_i) = S_{i+1}'(x_i) \quad i = 1, \ldots, n-1 \rightarrow n-1 \text{ equations}$$
$$S_i''(x_i) = S_{i+1}''(x_i) \quad i = 1, \ldots, n-1 \rightarrow n-1 \text{ equations}$$

$\Rightarrow$ $4n-2$ equations

$\Downarrow$

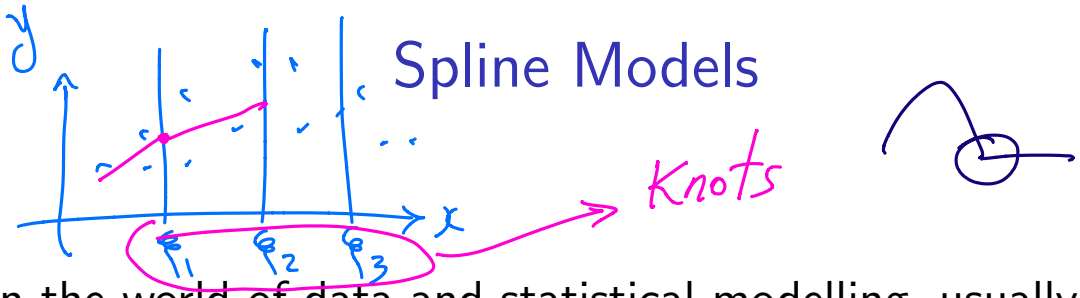
2 equations short:

$S_1''(x_0) = 0$

$S_n''(x_n) = 0$

linear fit beyond (x_0, x_n)

Spline Models



- In the world of data and statistical modelling, usually, we are not interested in a perfect fit!
- Consider a fixed set of knots $\xi_1, ..., \xi_K$.
- Spline approach could be applied to neighbourhoods with continuity/differentiability/smoothness constraints on neighbourhood boundaries, i.e. knots.
- A regression spline is a piecewise function explaining the behaviour of the data in each neighbourhood, subject to continuity (and possibly smoothness) conditions at the knots.

Order-M Spline

- An **order-M spline** with knots $\xi_1, ..., \xi_K$ is a piecewise polynomial of order M and has continuous derivatives up to order $M - 2$.

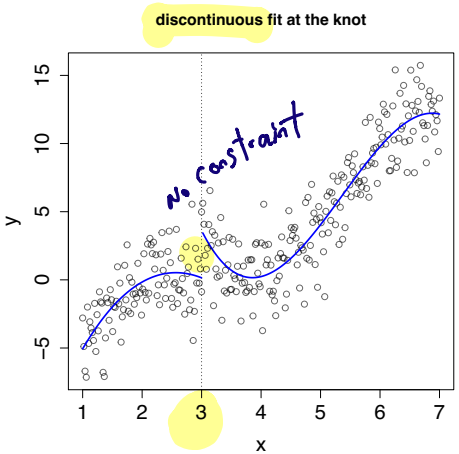
- Cubic splines (order-4) are the lowest-order spline for which the knot-discontinuity is not visible to the human eye. Therefore, there is seldom any good reason to go beyond cubic-splines.

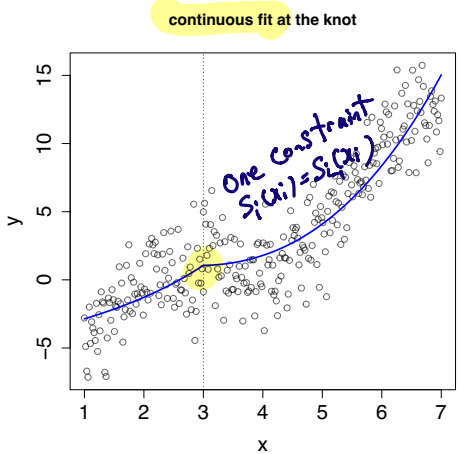
- In practice, the most widely used orders are $M = 1,\ 2,\ 4$: constant, linear, cubic.

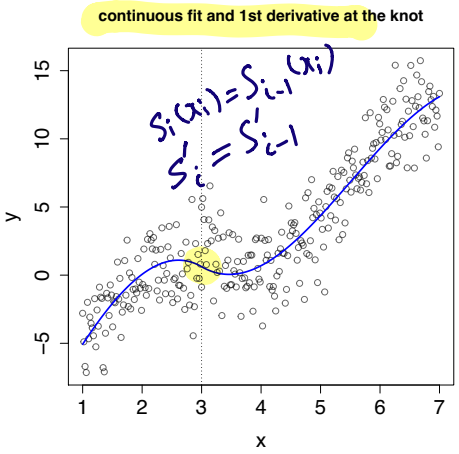
- A spline model with fixed knots is sometimes called a **regression spline**.

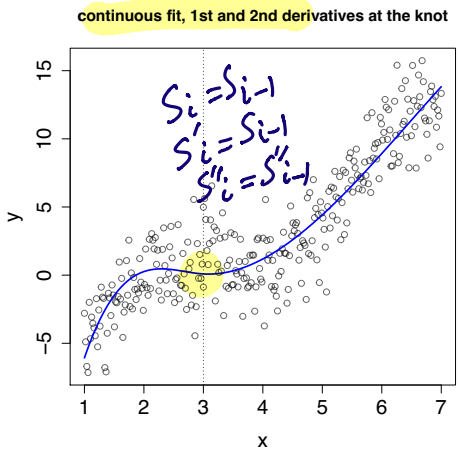
.

Piecewise Cubic Model







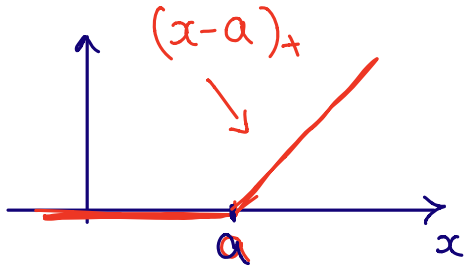


Cubic Splines

Definition: The positive part function $(x-a)_+$ is defined to be

$$(x-a)_+ = \max(0\ ,\ x-a)$$

.



Cubic Spline: Consider the fixed and known knots $\xi_1,...,\xi_K$, and consider the basis functions 1, x, x^2, x^3, $(x-\xi_i)^3_+$, $i=1,...,K$. A cubic spline model can be represented as

# of knots

$$\mu(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \sum_{j=1}^{K} \beta_{j+3}(x-\xi_j)^3_+$$

$$y = \mu(x) + \varepsilon$$

Cubic Splines : Remarks

For the representation in the previous slide, we have

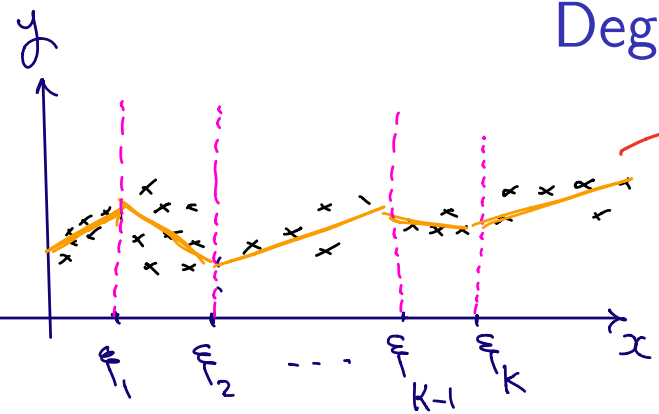
1) The model has $K + 4$ parameters.
   (this is called *the degrees of freedom of the model*)

2) $\mu(x)$ is continuous up to the second derivative.

Degrees of Freedom

- The **degrees of freedom of the model (df)** is the number of parameters required to fit the model.
- Usually, the larger the df, the more flexible the model.
- The top left panel of the plot in slide 27 has *too many* degrees of freedom. The constraints imposed on a cubic regression spline result in the more natural and smooth fit of the bottom-right panel.

Degrees of Freedom → # of param. in the model



→ Linear Spline

# of parameters	# of constraints :
$2 \times (K+1) = 2K+2$	$K \rightarrow 1$ for each knot

$\Rightarrow$ df of the model = # of parameters − # of constraints

$= 2K+2-K = K+2$

Cubic Spline :

of parameters $= (K+1) \times 4 = 4K+4$

# of constraints = K Continuity, K derivative, K 2nd derivative } 3K

$\Rightarrow$ df $= 4K+4-3K$

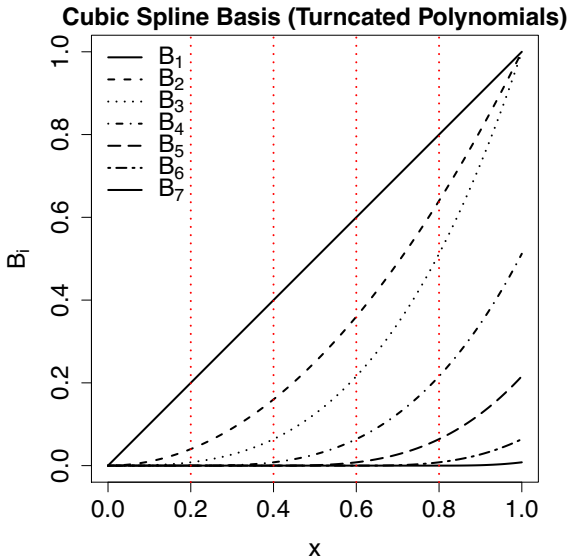
$= K+4$

K + order of the Polynomid

Cubic Splines

Cubic Splines: Consider the fixed and known knots $\xi_1,...,\xi_K$, and consider the basis functions 1, x, x^2, x^3, $(x-\xi_i)_+^3$, $i = 1, ..., K$. A cubic splines model can be represented as

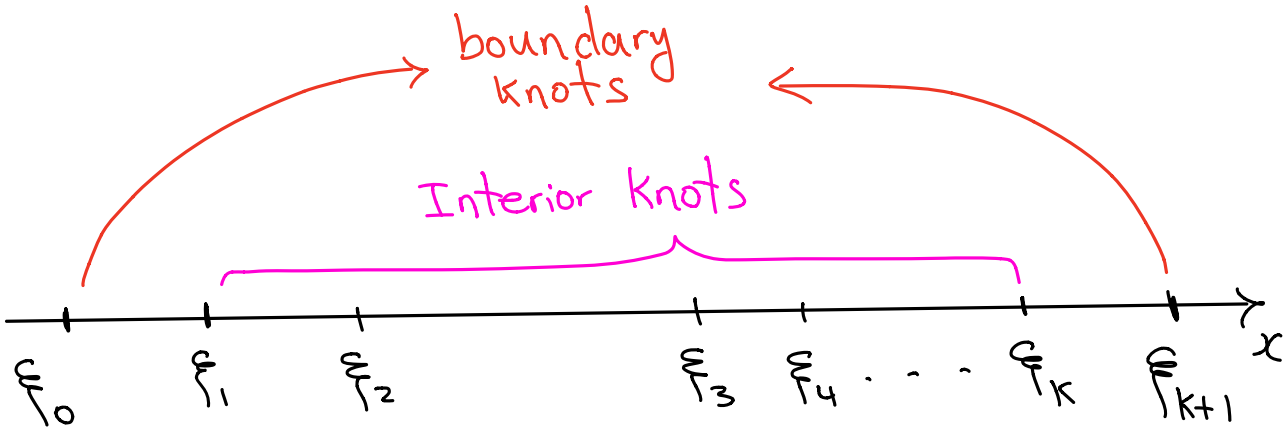
$$\mu(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \sum_{j=1}^{K} \beta_{j+3}(x-\xi_j)_+^3$$



Natural Cubic Splines

Natural Cubic Splines: here, we add boundary conditions to cubic splines: the function is required to be linear at the boundary, i.e. in the region smaller than the smallest knot of larger than the largest knot.

Natural splines (**don't have to be cubic**) provide more stable estimates at the boundaries.



Degrees of Freedom of Natural Cubic Splines

Assume $\xi_1, \dots, \xi_K$ are fixed.

$K-1$ neighbourhoods $\Rightarrow$ # of parameters $= 4(K-1)$

- continuity $\rightarrow$ $-K$
- first derivative $\rightarrow$ $-K$
- second derivative $\rightarrow$ $-K$

$K-4$

- before ξ_1 (linear): 2 parameters
- after ξ_K (linear): 2 parameters

$K-4+2+2 = K$ $\Leftarrow$

The df of the model is the # of knots

Basis Functions of Natural Cubic Splines

Considering the model $y = \mu(x) + \epsilon$, we are interested in modelling $\mu(x)$ with regression splines.

A natural cubic spline with K knots $\xi_1, ..., \xi_K$ is represented by K basis functions.

Imposing the linearity on the boundaries, and using cubic splines basis functions (with truncated power series), it can be show that

$$\mu(x) = \sum_{j=0}^{K-1} \beta_j N_j(x)$$

in which

$$N_0(x) = 1, \quad N_1(x) = x, \quad N_{k+1}(x) = d_k(x) - d_{K-1}(x)$$

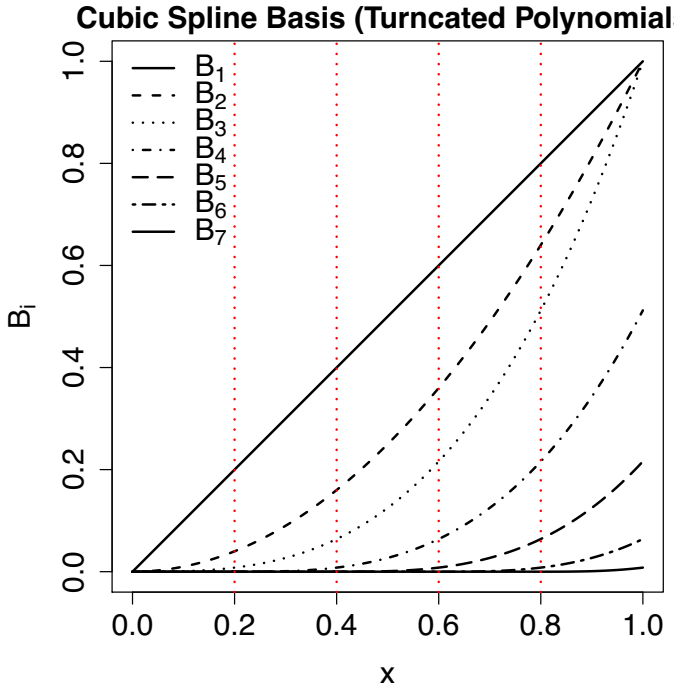
where

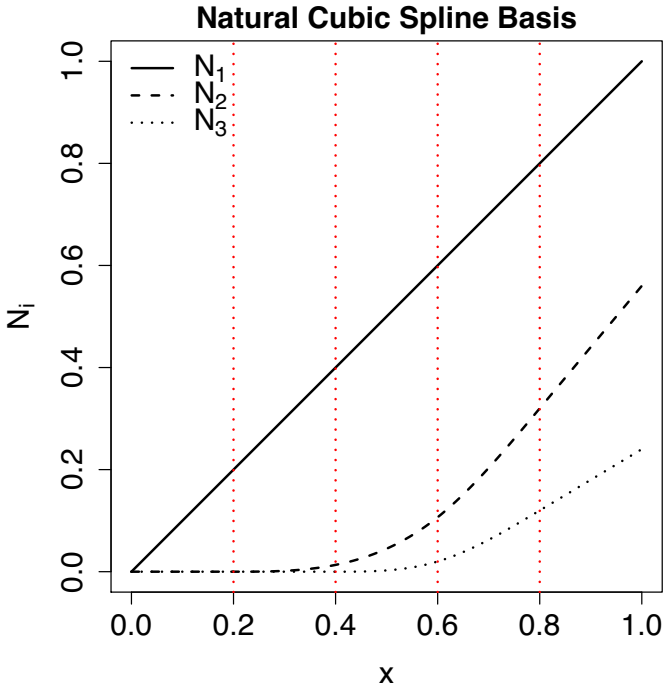
$$d_k(x) = \frac{(x - \xi_k)_+^3 - (x - \xi_K)_+^3}{\xi_K - \xi_k} \quad , \quad k = 1, ..., K-2$$

Basis Functions of Natural Cubic Splines

This rather unintuitive-looking basis has the nice property that the second and third derivatives of each N_j are zero outside the interval $(\xi_1 \ , \ \xi_K)$.

Basis Functions: Cubic Splines vs. Natural Cubic Splines





B-Splines

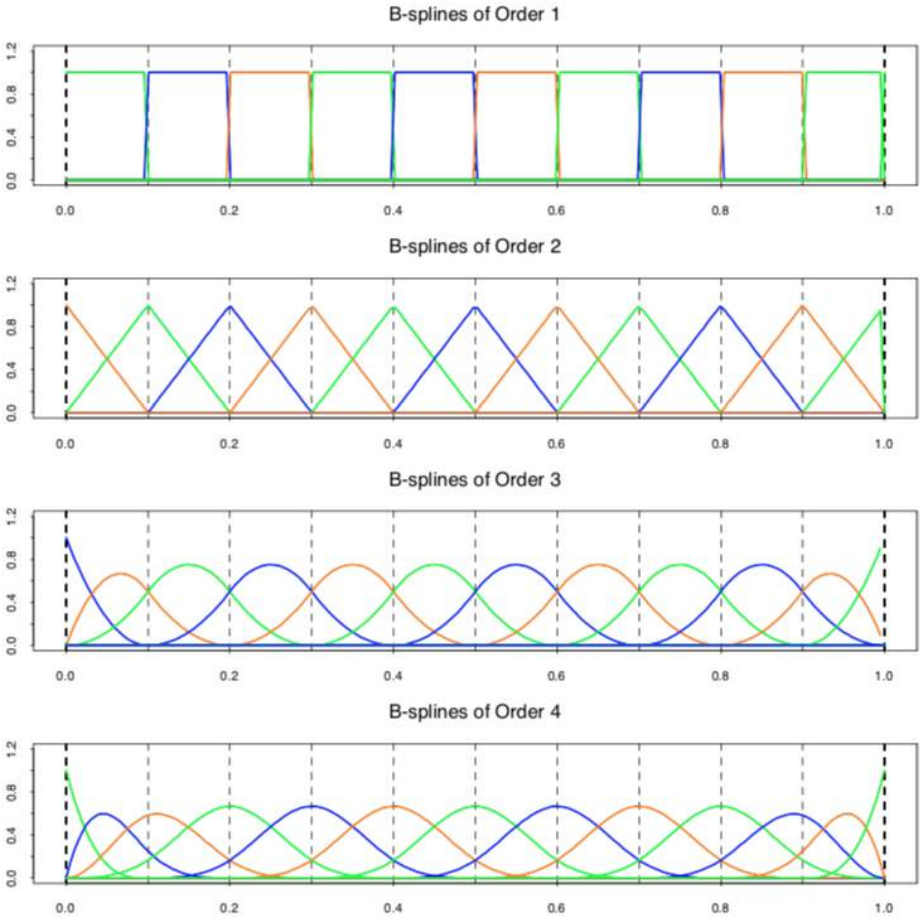
The basis functions based on polynomials and the positive part function is just one choice of basis functions.

This particular choice is fairly easy to understand conceptually, but it is, unfortunately, a poor choice for computation. The problem is that powers of large numbers can lead to numerical instability and round-off error.

Instead, an equivalent set of basis functions but one which is better computationally is the so-called **B-spline basis**, which also allows for efficient computation even when K, the number of knots, is large.

We will not discuss the details but refer to *Elements of Statistical Learning* by Hastie, Tibshirani and Friedman (pp. 186–189).

B-Splines or order m



constant

linear

quadratic

cubic

FIGURE 5.20. *The sequence of B-splines up to order four with ten knots evenly spaced from 0 to 1. The B-splines have* local support*; they are nonzero on an interval spanned by $M + 1$ knots.*

Fitting Natural Cubic Splines

Consider the model $y = \mu(x) + \epsilon$ where we assume

$$\mu(x) = \sum_{j=0}^{K-1} \beta_j N_j(x)$$

then

$$y = \sum_{j=0}^{K-1} \beta_j N_j(x) + \epsilon$$

which is a regular multiple linear regression problem with design matrix generated by the basis functions N_j, $j = 0, ..., K-1$.

$\underset{\sim}{Y} = N\underset{\sim}{\beta} + \underset{\sim}{\varepsilon}$ where

$$N = \begin{pmatrix} N_1(x_1) & \cdots & N_K(x_1) \\ N_1(x_2) & \cdots & N_K(x_2) \\ \vdots & & \\ N_1(x_n) & \cdots & N_K(x_n) \end{pmatrix}$$

$$\hat{\beta} = (N^T N)^{-1} N^T Y$$

$$\hat{Y} = N\hat{\beta}$$

Robust Regression

$\min \sum_i w_i (y_i - \hat{y}_i)^2$ (w_i: weight)

Fitting Natural Cubic Splines and B-Splines in R Software

In `splines` package a basis matrix is provided by two functions

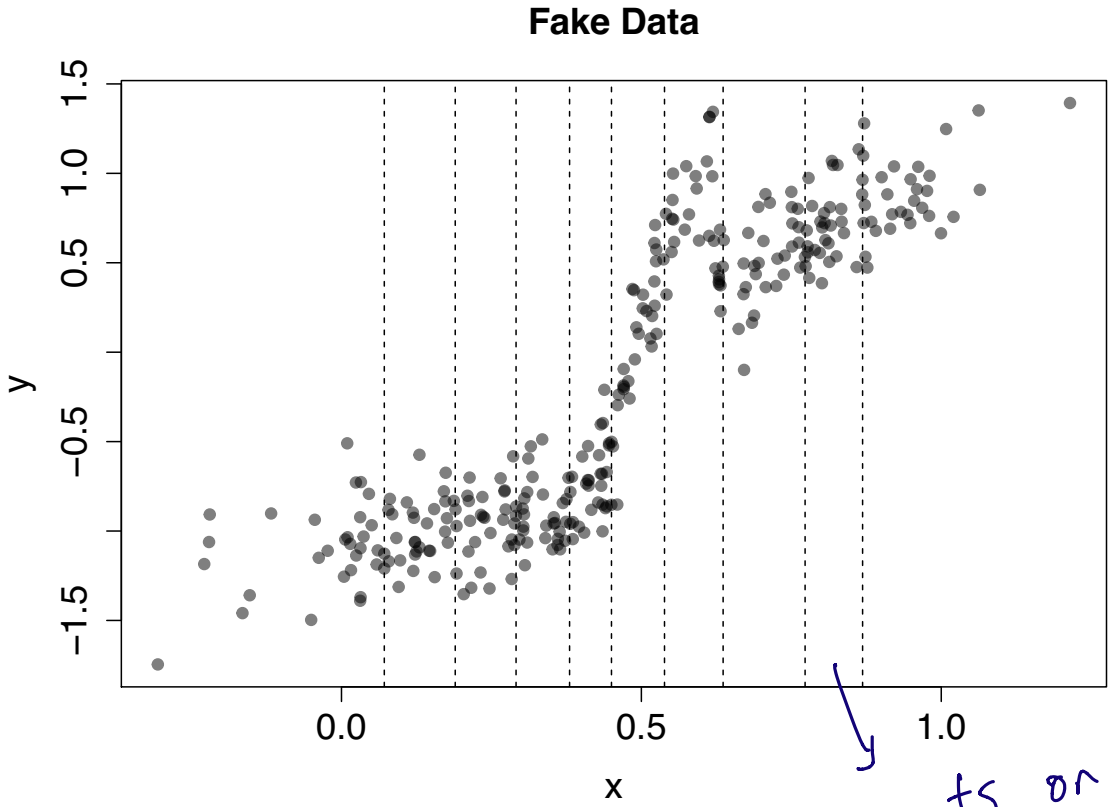
- `ns(...)` and only for natural cubic splines.
- `bs(...)` for B-splines

Cubic polynomials are by far the most common and, as it turns out, are all we really need in most circumstances.

In R, in addition to any **interior knots** provided by the user (or determined automatically from user supplied arguments) two additional **boundary knots**, ξ_0 and ξ_{K+1} may be supplied ($\xi_0 < \xi_1$ and $\xi_{K+1} > \xi_K$). These determine the points beyond which the lower degree polynomials (here linear) are fit. By default, `ns` chooses the boundary knots at the minimum and maximum x values in the data.

Simulated Data → 300 data points

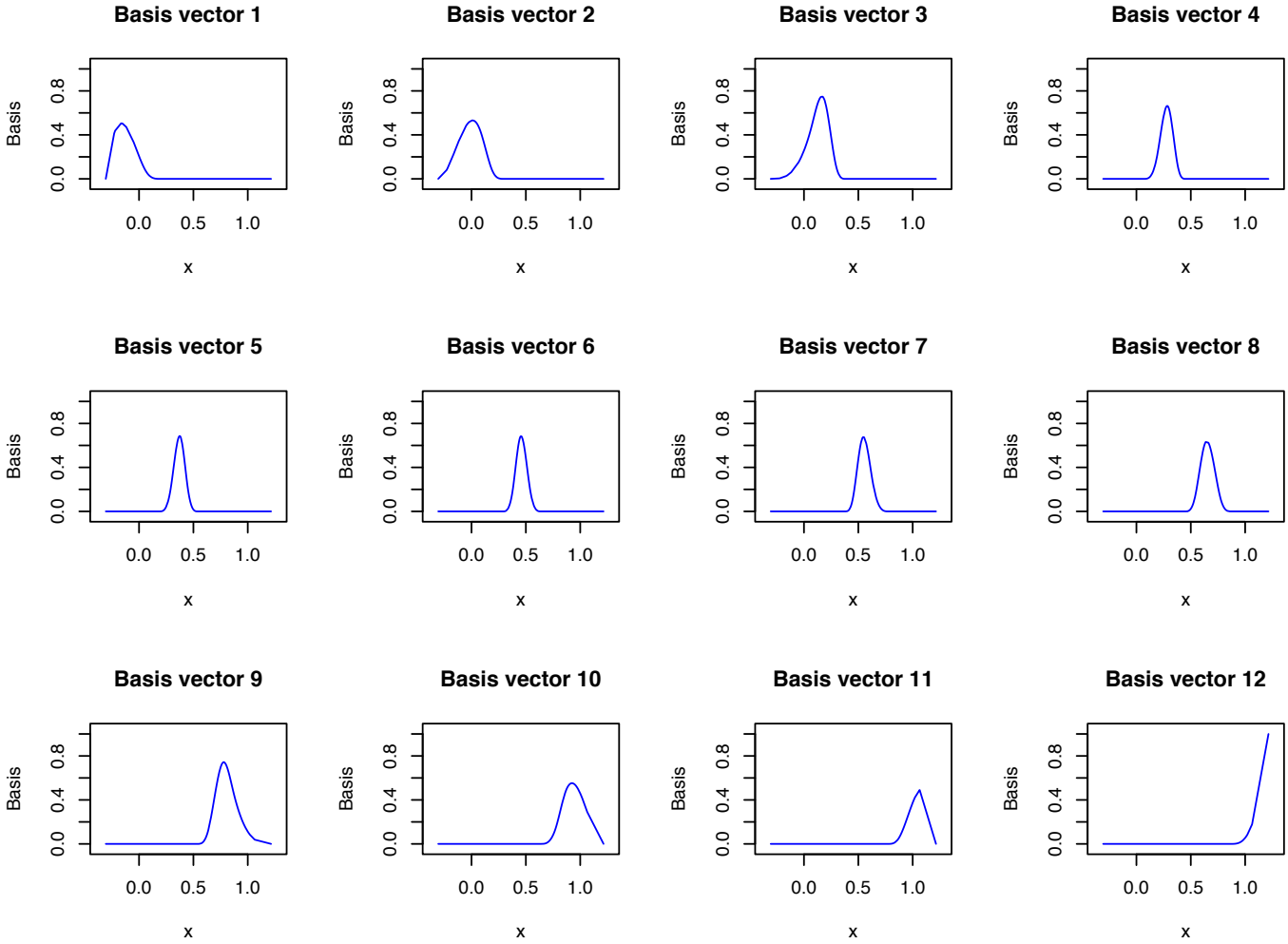
Consider the simulated data below:



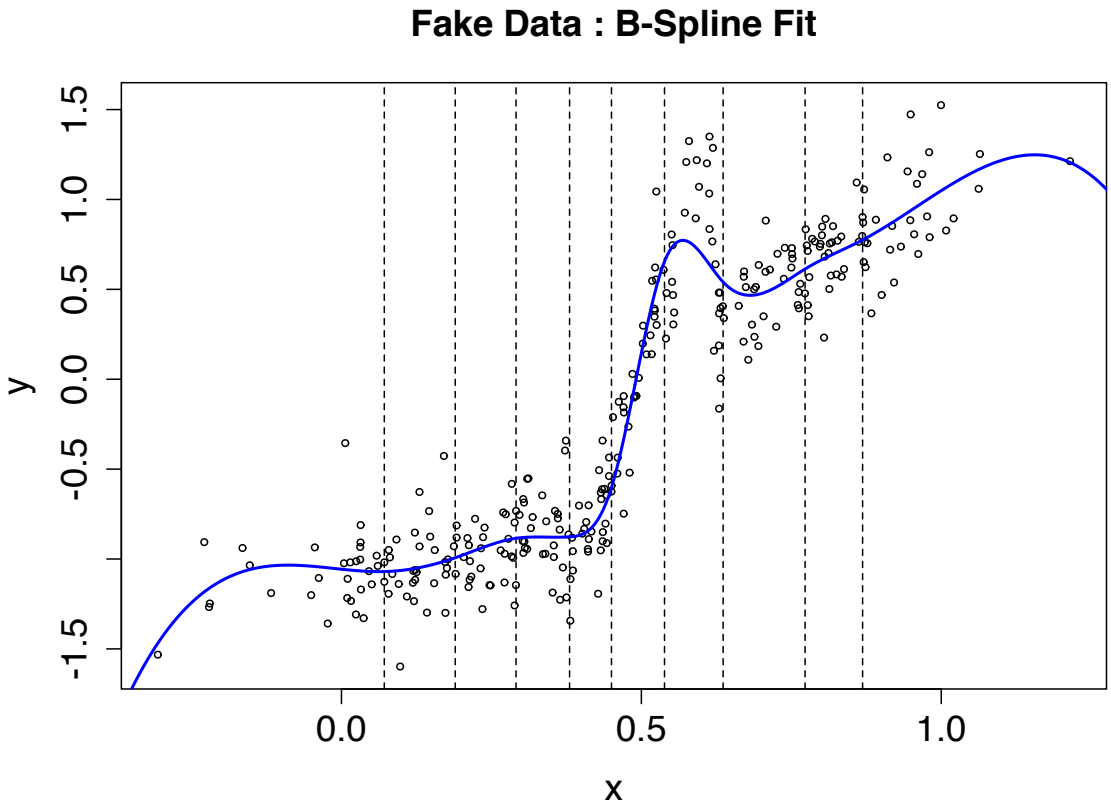
Knots on the quantiles of x $Q_{0.1}, Q_{0.2}, \ldots, Q_9$

Simulated Data : B-Spline Basis Functions

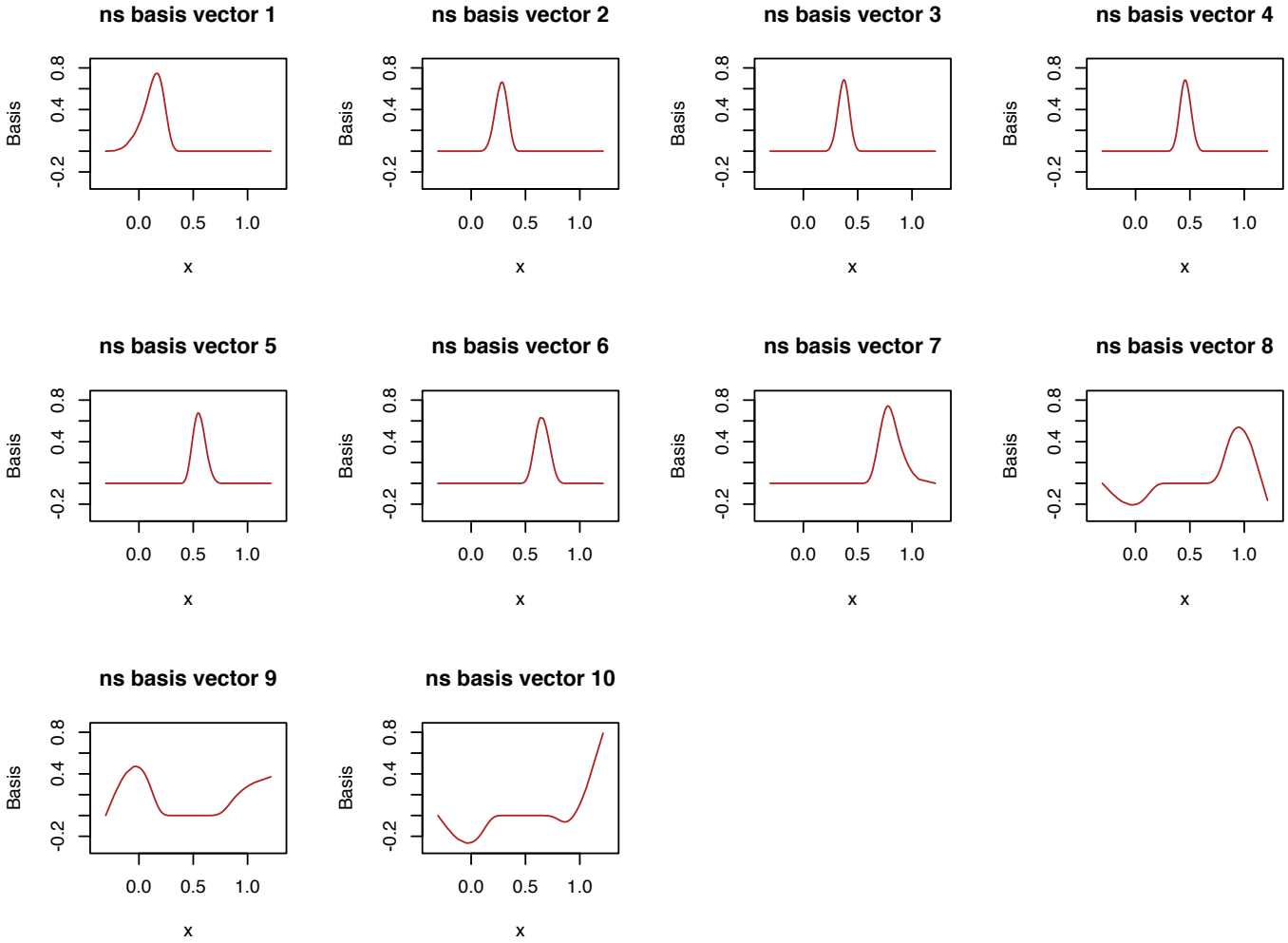
degree 3 (cubic)



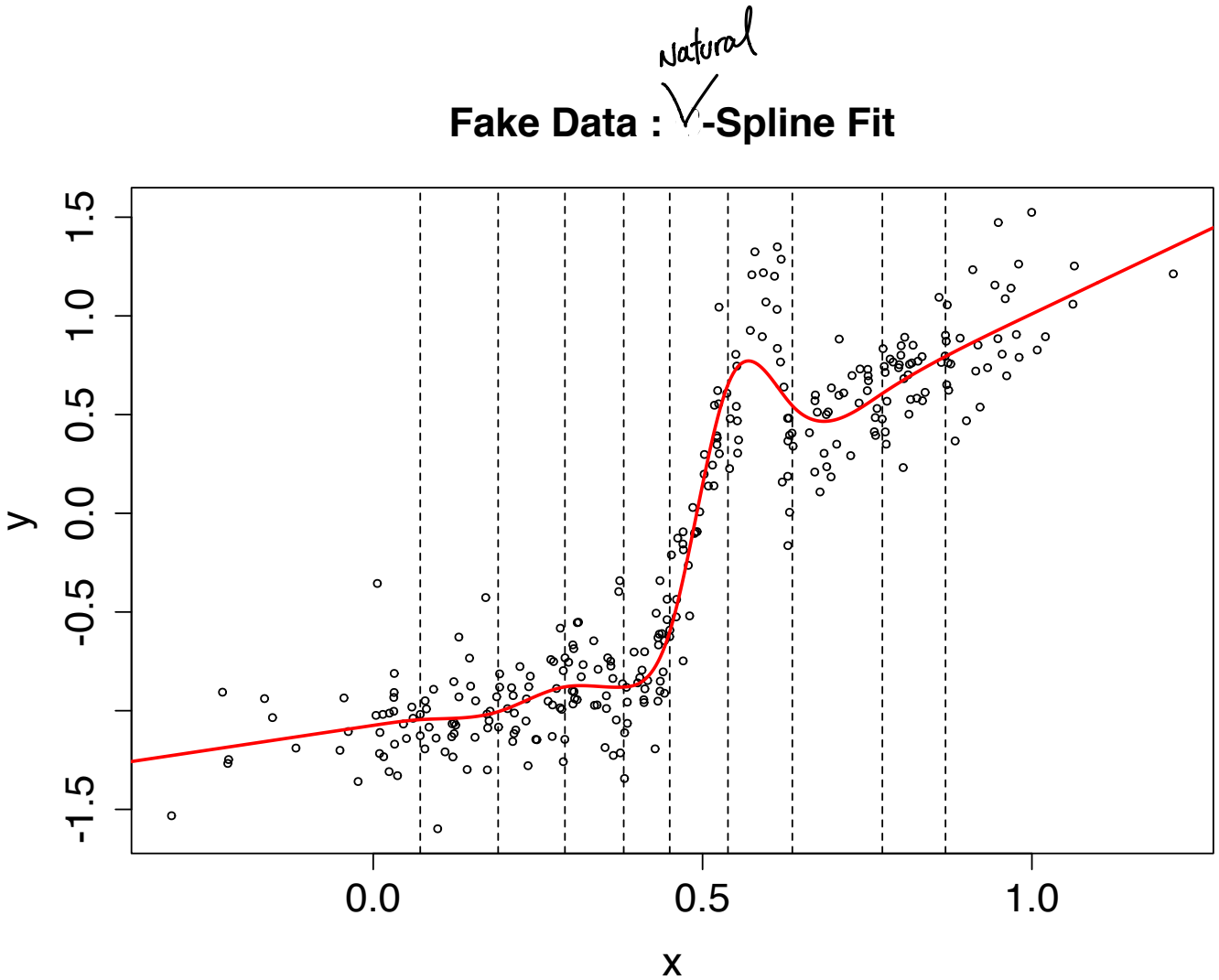
Simulated Data : B-Spline Fit



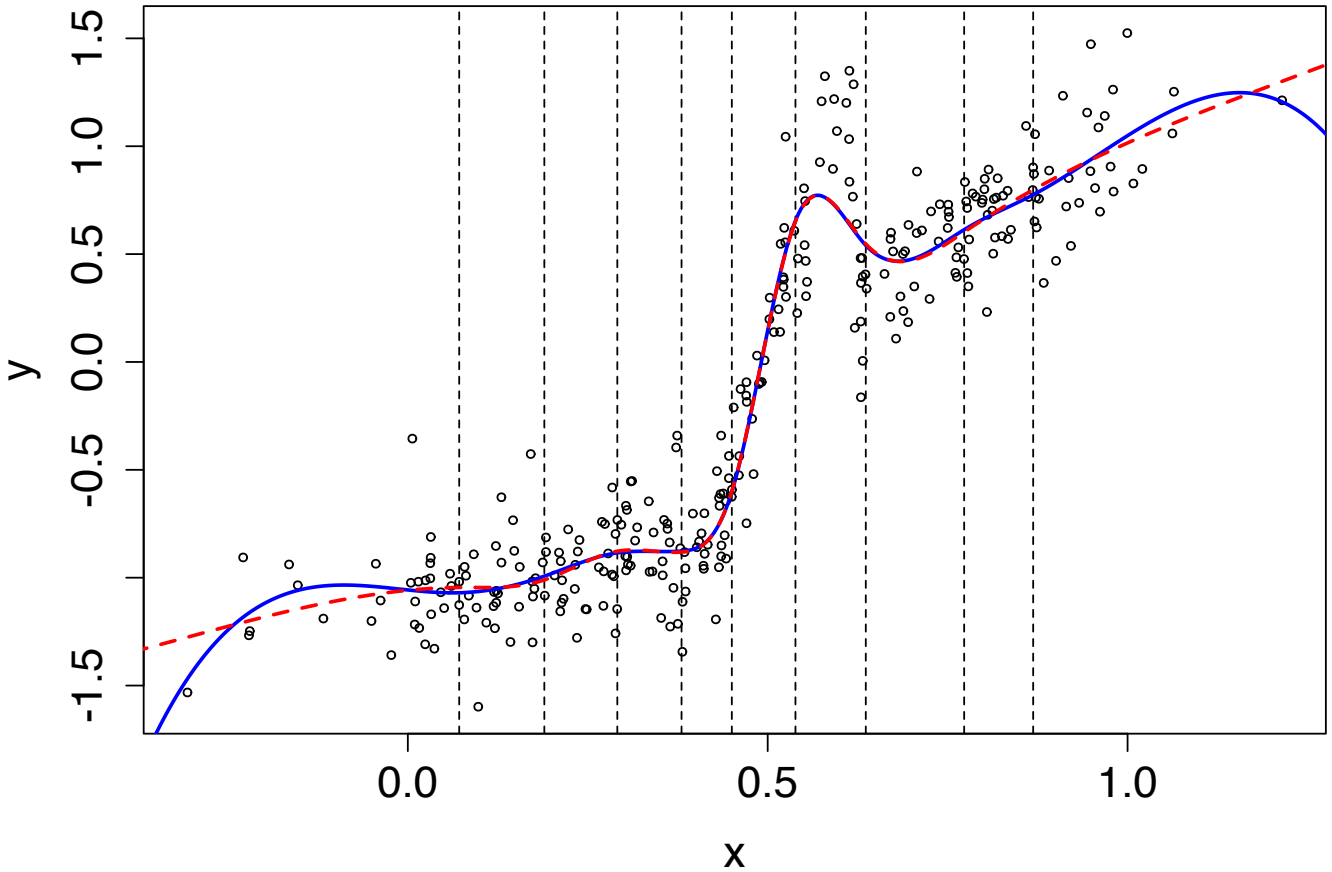
Simulated Data : Natural Basis Functions



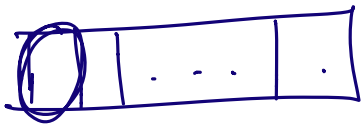
Simulated Data : Natural Spline Fit



Simulated Data : Natural Spline vs. B-Spline



Location of the Knots



Given K number of knots, where should we place the knots?

The regression spline is most flexible in regions that contain a lot of knots, because in those regions the polynomial coefficients can change rapidly.

It is reasonable to put more knots where the function changes more rapidly and to put fewer knots where it seems stable.

In practice, it is common to place the knots in a uniform fashion.

Another method is to estimate the location of the knots. For example, DiMatteo *et al.* (2001) used the reversible jumps MCMC to estimate both the number of location of the knots within a computational neuroscience context.

Varying Degrees of Freedom

The degrees of freedom depends on

1) number of knots
2) degree of the local polynomial (in practice: 0, 1, or 3)

Model selection methods and cross-validation can be used to choose the degree of freedom.

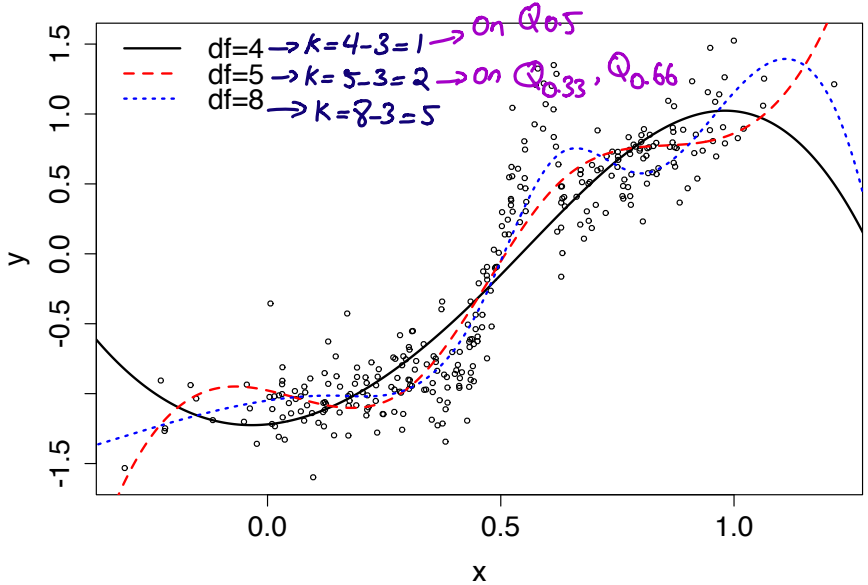
Varying Degrees of Freedom

`bs(x, degree= p, df=df)`: here `R` will put $df - p$ knots uniformly on the quantiles of x.

The `df` argument in the `bs` function is different from the degrees of freedom which was defined on slide 34.

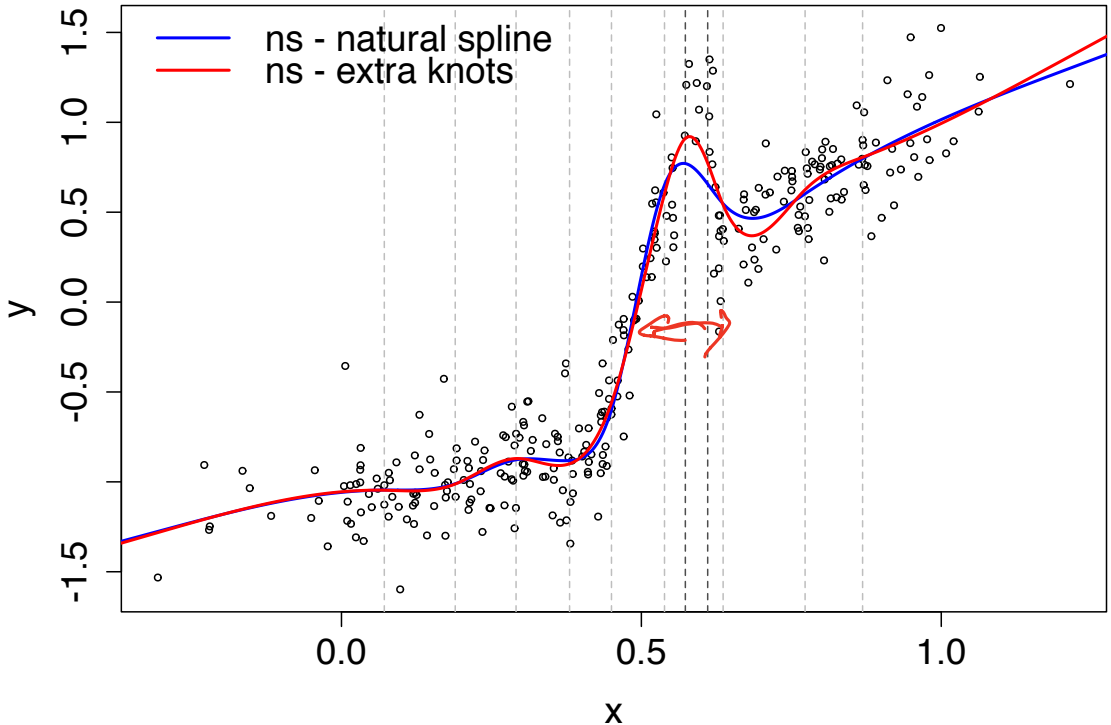
Here we are fitting cubic splines $(p = 3)$ with varying $df = 4, 5, 8$

$p + k = df \Rightarrow k = df - p$ → # of knots



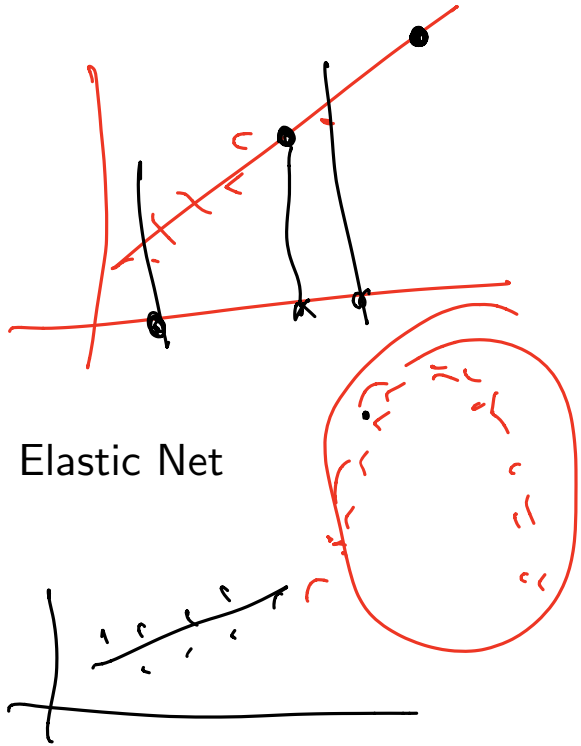
Varying Degrees of Freedom

You may decide to put more knots where there might be more variability in the function.



Agenda

- Introduction to Smoothing Methods
  - Polynomial Regression ✓
  - Basis Expansion ✓
- Splines
  - Cubic, Natural, and B-splines ✓
- Regularization Methods
  - Shrinkage Estimators: Ridge, LASSO, Elastic Net
  - Smoothing Splines
- Local Weighting
  - ~~Delta Neighbourhood and KNN~~
  - Kernel Smoothing
  - Local Smoothing and LOESS
- Multidimensional Smoothing (if time permits)
  - Tensor Product Bases
  - Thin-plate Splines
  - Additive Spline Model



Some Model Selection Criteria

1. R^2 and adjusted R^2,
2. Akaike's Information Criterion (AIC, AICs),
3. Bayesian Information Criterion (BIC),
4. Stepwise Method in Variable Selection,
5. PRESS-type criteria → prediction error

More modern modelling techniques involve regularization methods such as *LASSO*, *ridge* regression, and *elastic net*, some of which incorporate variable selection in the process of parameter estimation.

Model Selection Techniques

The common methods of model selection are the following:

- Least squares fit
- Subset Selection
- Shrinkage
- Dimension Reduction

Model Selection Techniques

Introduction to Statistical Learning by Gareth *et al.* (2013, p.204) defines the last three methods as follows:

1) **Subset Selection:** This approach involves identifying a subset of the p predictors that we believe to be related to the response. We then fit a model using least squares on the reduced set of variables.

Model Selection Techniques

$\beta_1, \ldots, \beta_p$ $\quad \sum (y_i - \hat{y}_i)^2$

2) **Shrinkage:** This approach involves fitting a model involving all p predictors. However, the estimated coefficients are shrunken towards zero relative to the least squares estimates. This shrinkage (also known as regularization) has the effect of reducing variance. Depending on what type of shrinkage is performed, some of the coefficients may be estimated to be exactly zero. Hence, shrinkage methods can also perform variable selection.

Model Selection Techniques

3) **Dimension Reduction:** This approach involves projecting the p predictors into a M-dimensional subspace, where $M < p$. This is achieved by computing M different linear combinations, or projections, of the variables. Then these M projections are used as predictors to fit a linear regression model by least squares.

Shrinkage Methods

We will discuss the following three shrinkage methods in variable selection for regression:

- Ridge regression
- LASSO
- Elastic net

All of these methods fall under shrinkage or *regularization* methods.

$$RSS(\lambda) := \underbrace{\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2}_{\text{Loss Function}} + \underbrace{\lambda \times \text{pen}(\boldsymbol{\beta})}_{\text{Penalty}}$$

Ridge Regression

Definition: The ridge regression estimate of a linear model is defined as

$$\widehat{\boldsymbol{\beta}}^{ridge} := \arg\min_{\boldsymbol{\beta}} RSS(\lambda),$$

L2 penalty

where

$$RSS(\lambda) := \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

where $\lambda > 0$ is a tuning parameter which determines the bias-variance trade-off. This problem can be written in an equivalent form as

$$\widehat{\boldsymbol{\beta}}^{ridge} := \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \quad \text{s.t.} \quad \sum_{j=1}^{p} \beta_j^2 \leq t$$

where there is a one-to-one correspondence between λ and t.

Ridge Regression (cont'd)

Note that

$$RSS(\lambda) := \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

$$:= (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^T \boldsymbol{\beta}$$

Calculating the derivative of $RSS(\lambda)$ w.r.t. the vector $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)$, setting it to zero, and solving the equation, we get (show this):

$$\widehat{\boldsymbol{\beta}}^{ridge} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}$$

where I is the $p \times p$ identity matrix.

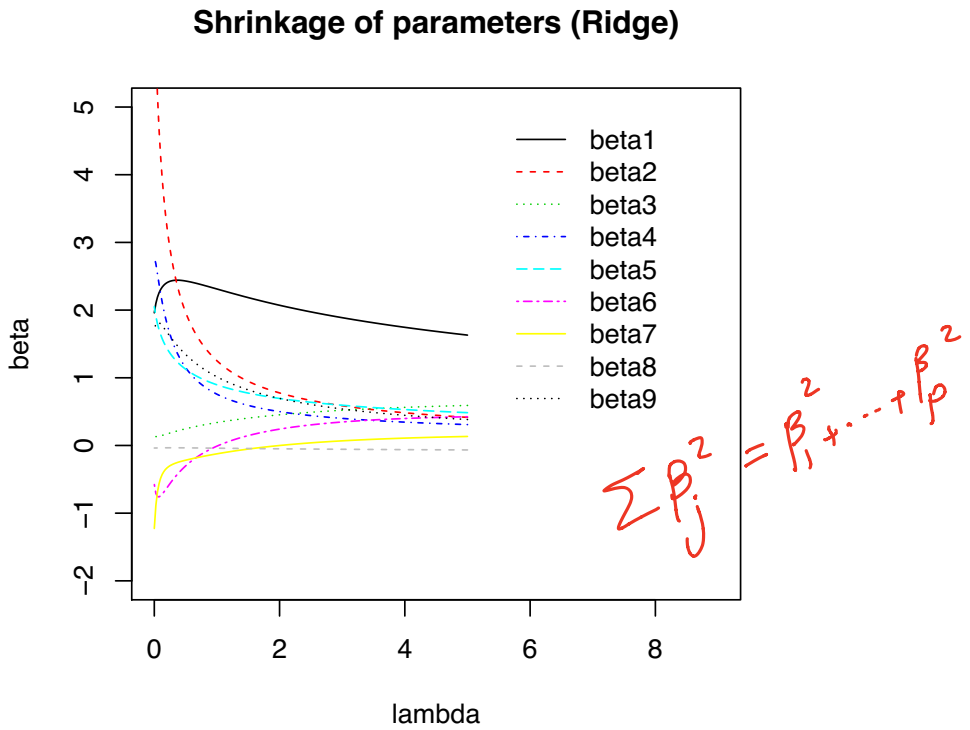
It turns out that $\widehat{\beta}_0 = \overline{y} - \sum_{j=1}^{p} \overline{x}\widehat{\beta}_j$, so centralizing the columns of $\mathbf{X}$ (i.e. $x_{ij} - \overline{x_j}$, $j = 1., .., p$) results in $\widehat{\beta}_0 = \overline{y}$. Centralizing both $\mathbf{y}$ (i.e. $\overline{y}_i - \overline{y}$) and the columns of $\mathbf{X}$ results in $\widehat{\beta}_0 = 0$.

Example

The market value of a house should be a function of a number of features of the house and a model for sale price as a function of these features can be useful.

X_1	Current taxes
X_2	Number of Bathrooms
X_3	Lot size
X_4	Living space
X_5	Number of parking spaces
X_6	Number of rooms
X_7	Number of bedrooms
X_8	Age of house
X_9	Number of fireplaces
Y	Actual sale price

House Price Data-set: Ridge Regression



Note that the some of the parameter estimates get very close to 0, but they are not exactly 0, hence ridge regression does not perform a direct variable selection.

LASSO

Ridge : $\sum_{j=1}^{p} \beta_j^2$

Definition: The LASSO regression estimate of a linear model is defined as

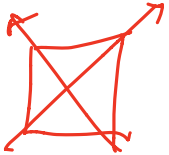
$$\widehat{\boldsymbol{\beta}}^{lasso} := \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

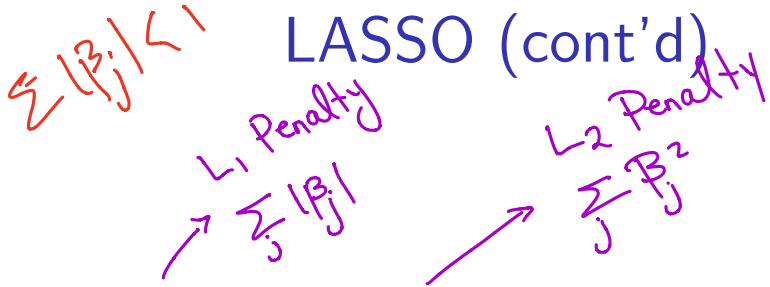
where $\lambda > 0$ is a tuning parameter which determines the bias-variance trade-off. This problem can be written in an equivalent form as

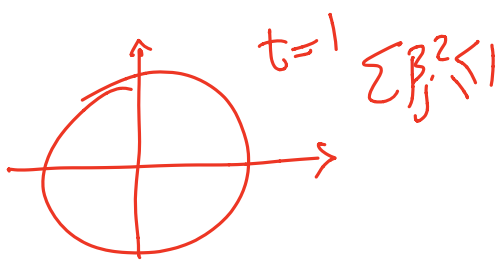
$$\widehat{\boldsymbol{\beta}}^{lasso} := \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \quad \text{s.t.} \quad \sum_{j=1}^{p} |\beta_j| \leq t$$

where there is a one-to-one correspondence between λ and t.

LASSO (cont'd)

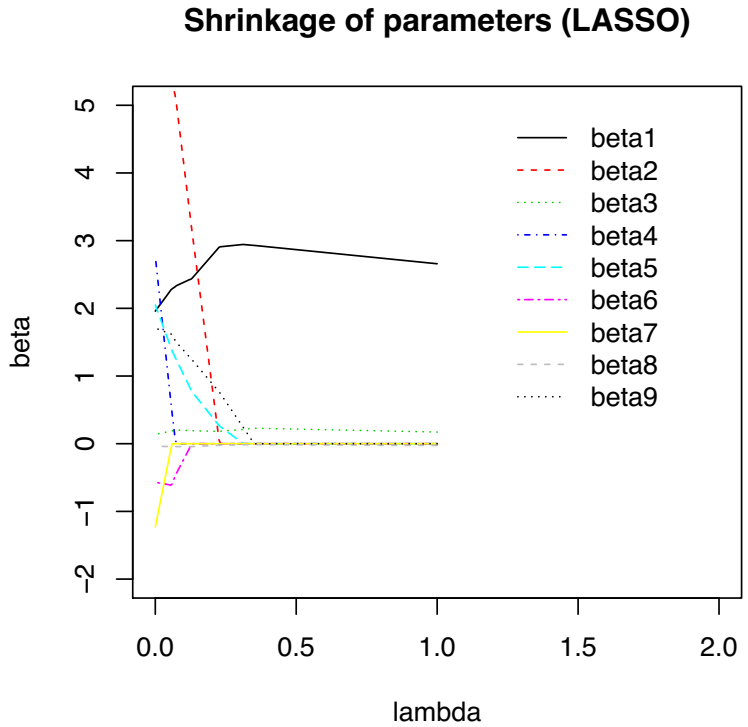


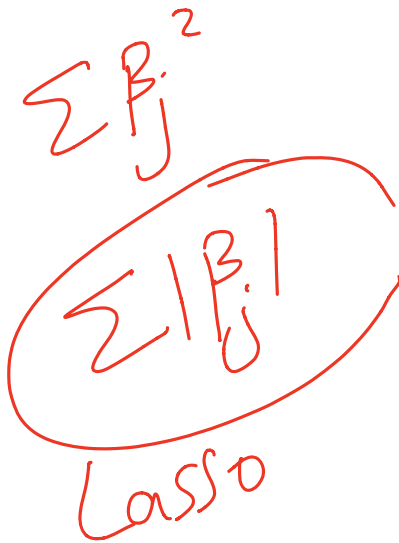




- Comparing LASSO to ridge regression, we see that the constraint is on the absolute value of the parameter values, not the squares.
- Unlike ridge regression, LASSO does not have a closed form solution for the parameter estimates $\widehat{\beta}^{lasso}$.
- The L_1 penalty used in LASSO works as a variable selection as the shrinkage forces some of the parameter estimates to be exactly 0.

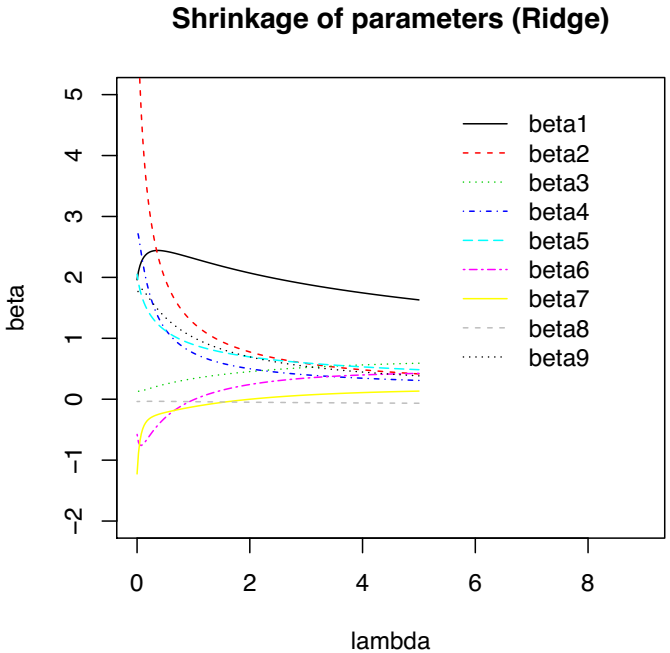
House Price Data-set: LASSO

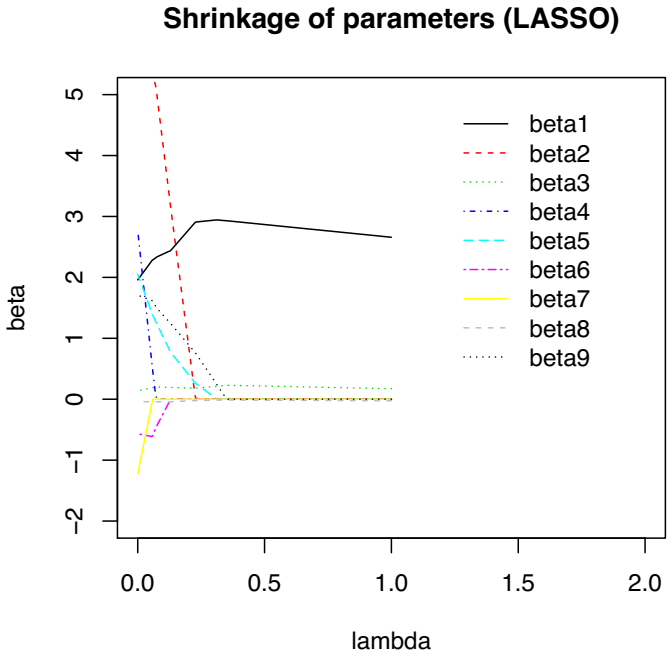




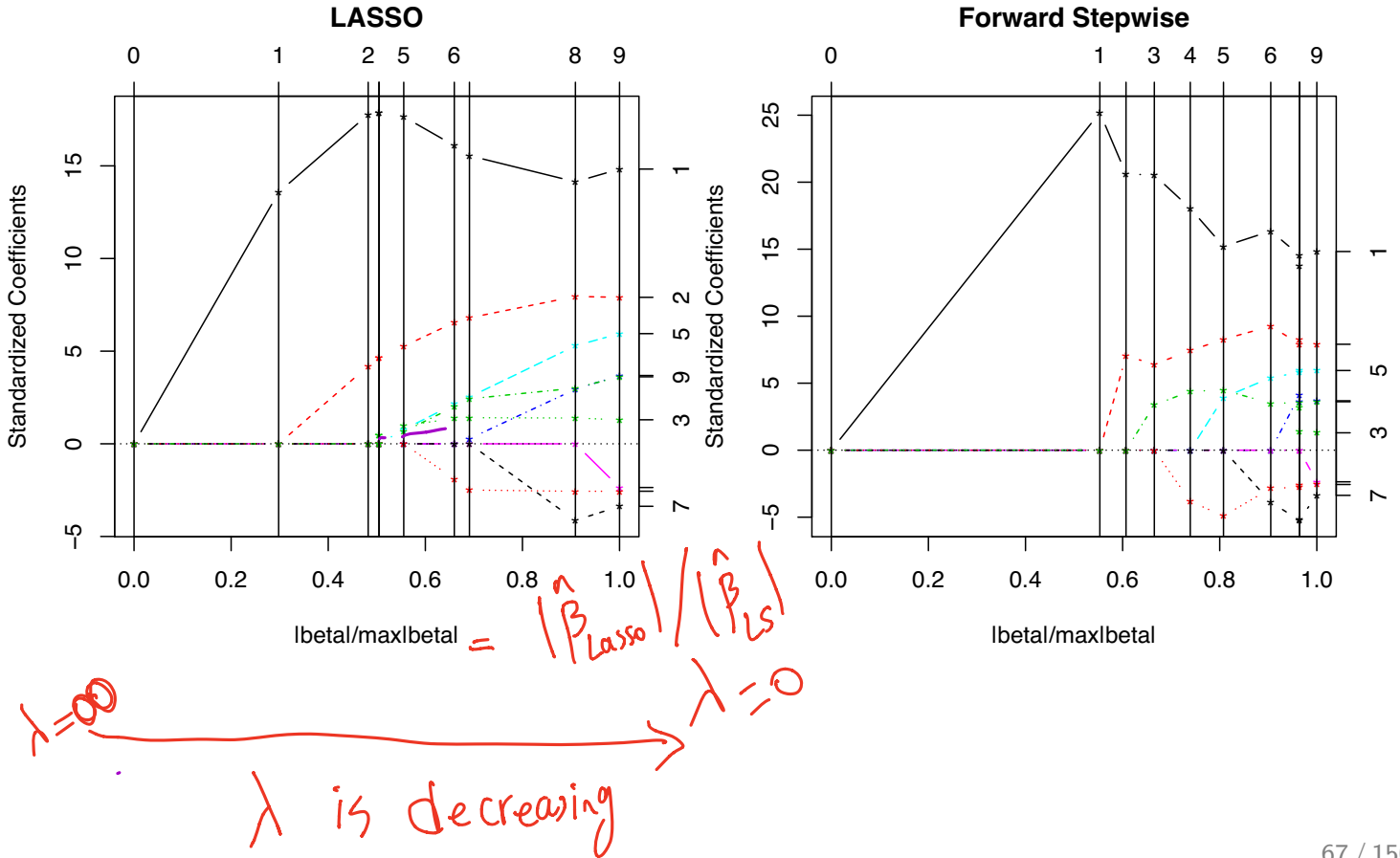
Note that as the penalty λ increases, the parameter estimates are shrunk to 0, hence Lasso performs a direct variable selection.

Ridge vs. LASSO





LASSO and Variable Selection



Variable Selection Using LASSO (from ISLR Book)

Consider a regression model with two explanatory variates, hence parameter vector $\boldsymbol{\beta} = (\beta_1, \beta_2)$.

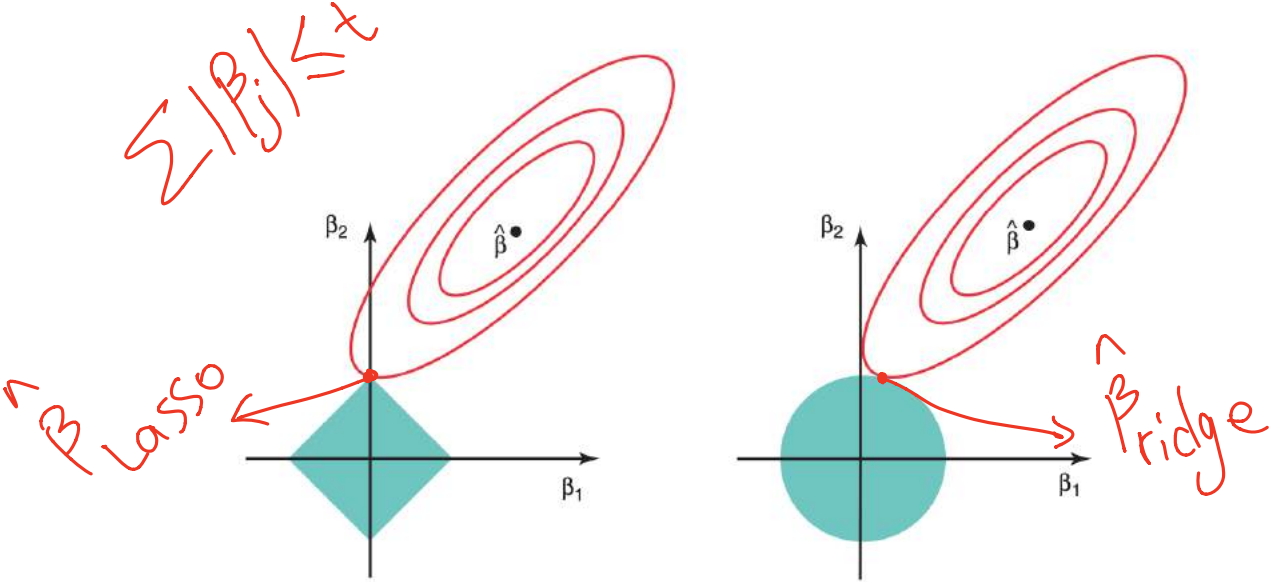


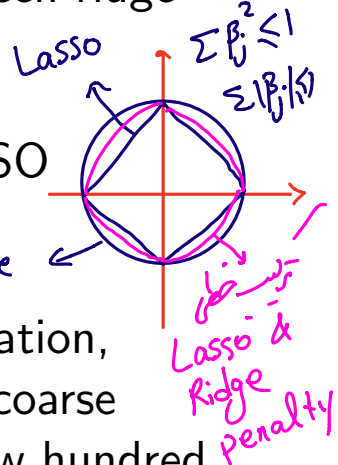
FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \le s$ *and* $\beta_1^2 + \beta_2^2 \le s$, *while the red ellipses are the contours of the RSS.*

Elastic Net Regression

Definition: The elastic net regression estimate of a linear model is defined as

$$\widehat{\beta}^{EN} := \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \left[(1-\alpha)\beta_j^2 + \alpha|\beta_j| \right] \right\}$$

where $\lambda > 0$ determine the overall complexity of the model, and the elastic net parameter $\alpha \in [0, 1]$ provides a mix between ridge regression and the lasso.



The package `glmnet` is used for Ridge (`alpha=0`), LASSO (`alpha=1`), and Elastic Net ($\alpha \in (0, 1)$).

While parameters α and λ can be chosen via cross-validation, typically a grid search is used. The grid for α is usually coarse (three to five values) while that of λ is much finer (a few hundred values).

More Details on Lasso, Ridge, and Elastic Net

- ISLR : Chapter 6 (6.2, and ideas from 6.4, and 6.6)
  - Videos by the authors available in this link.
- ESL : Chapter 3 (3.4)
- Statistical Learning with Sparsity

There are variations of LASSO (grouped LASSO, fused LASSO, etc.) which are beyond the cope of this short course. See the last reference above for details.

Next, we take the regularization idea to spline models, i.e. *smoothing splines.*

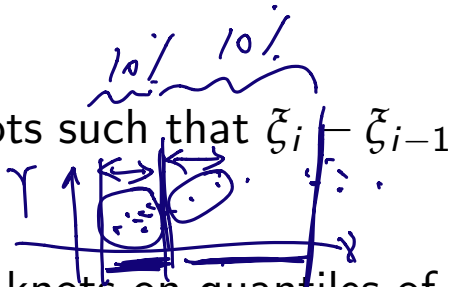
Agenda

$$\min_{\beta} \left[\sum (y_i - \hat{y}(x))^2 + \lambda \, \text{pen}(\lambda) \right]$$

- Introduction to Smoothing Methods
  - Polynomial Regression ✓
  - Basis Expansion ✓
- Splines
  - Cubic, Natural, and B-splines ✓
- Regularization Methods
  - Shrinkage Estimators: Ridge, LASSO, Elastic Net ✓
  - Smoothing Splines
- Local Weighting
  - ~~Delta Neighbourhood and KNN~~
  - Kernel Smoothing
  - Local Smoothing and LOESS
- Multidimensional Smoothing (if time permits)
  - Tensor Product Bases
  - Thin-plate Splines
  - Additive Spline Model

Location/Number of Knots Splines

Whether B-Spline or natural spline is used, we still need to decide on the number and locations of the knots.



- *Equal-distance knots:* Distribute the knots such that $\xi_i - \xi_{i-1}$ is constant $i = 2, ..., k$.
- *Equal-distance quantiles:* Distribute the knots on quantiles of the data. One choice to distribute k knots is to locate them at $\frac{i}{k+1}$, $i = 1, ..., k$ quantiles.
- *Corss-validaiton:* following a regime for location of the knots, e.g. equal-distance quantiles, you may use cross-validation to choose the number of knots.

Smoothing Splines

$Y|X = \mu(X) + \varepsilon$

Knot selection can be avoided by recasting the problem. Note that we would still like to use the Splines.

We would like to estimate the function $\mu(x)$ which minimizes $RSS = \sum_{i=1}^{n}(y_i - \mu(x_i))^2$, but that is also *smooth*.

We can let the data "smooth itself," i.e. control the smoothness of the fit by *regularization*. In other words, we will penalize the *RSS* for overfitting or roughness.

In fact, rather than defining smoothness (which is what we want), we will define what we don't want, namely a non-smooth or "rough" function. But how to define "rough?"

A Measure of Roughness

- Suppose $\mu(x)$ is, at least, twice differentiable.
- $\mu'(x)$ measures the slope. A function with any slope can be smooth.
- A function with frequent or abrupt changes in the slope though would be rather rough.
- The second derivative $\mu''(x)$ measures how quickly the slope changes at any given point x.
- Large values of $(\mu''(x))^2$ indicate that there is an abrupt change in slope at the point x. Therefore, one possible measure of roughness might be

$$\int (\mu''(t))^2 \, dt$$

The smaller is the integral, the smoother is $\mu(x)$.

Smoothing Splines

A smoothing spline $\widehat{\mu}(x)$ is a function estimate of $\mu(x)$ obtained by minimizing

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

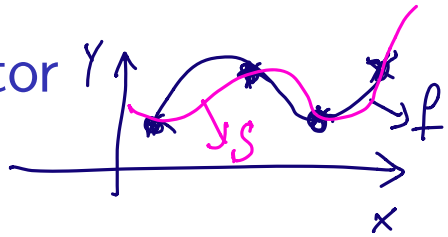
roughness

where $\lambda \geq 0$ is a fixed *smoothing parameter* (also called a tuning parameter).

Effect of λ:

- $\lambda = 0$: no roughness penalty $\Rightarrow$ perfect fit $\mu(x_i) = y_i \quad i = 1, 2, \ldots, n$
- $\lambda = \infty$: OLS estimate (least squares) for linear model $y_i = \beta_0 + \beta_1 x_i$ $i = 1, \ldots, n$
- Lagrange multiplier?
  - has the same flavour.

The "Smoothest" Interpolator



Theorem: Suppose $f(x) : \mathbb{R} \to \mathbb{R}$ is a real function whose value is known only at a set of n distinct points $x_1, ..., x_n$. The points $(x_1\ ,\ f(x_1))$,..., $(x_n\ ,\ f(x_n))$ can be used to determine natural cubic splines $s(x)$ such that $s(x_i) = f(x_i)$, $i = 1, ..., n$. For any differentiable function $g(x)$ passing through the points $(x_i\ ,\ f(x_i))$, we have

$$\int_{-\infty}^{\infty} \left(s''(x)\right)^2\ dx \leq \int_{-\infty}^{\infty} \left(g''(x)\right)^2\ dx$$

This means that the NCS is the "smoothest" interpolator.

Smoothing Splines and Natural Cubic Splines

For any fixed λ, the solution to the penalized residual sum of squares problem

$$\min_{\mu} \left[\sum_{i=1}^{n} (y_i - \mu(x_i))^2 + \lambda \int (\mu''(t))^2 \, dt \right]$$

is **a natural cubic spline with knots at every unique value of** x_i.

In other words, the solution requires $\widehat{\mu}(x)$ to be of the form

$$\widehat{\mu}(x) = \sum_{j=1}^{n} N_j(x) \widehat{\beta}_j$$

where the $N_j(x)$, $j = 1, ..., n$ is a set of n basis functions for this family of natural cubic splines.

Deriving $\widehat{\beta}$ (better notation : $\widehat{\beta}_\lambda$)

$$\mu(x) = \sum_{j=1}^{n} \beta_j N_j(x) \qquad \widehat{Y} = N\widehat{\beta}$$

$$\widehat{\beta} = \underset{\mu}{\operatorname{argmin}} \left[\sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{n} \beta_j N_j(x) \right)^2 + \lambda \int_{-\infty}^{+\infty} \left(\sum_{j=1}^{n} \beta_j N_j''(x) \right)^2 dx \right]$$

$$\underbrace{(\underset{\sim}{y} - N\underset{\sim}{\beta})^T (\underset{\sim}{y} - N\underset{\sim}{\beta})}$$ where $N = \begin{pmatrix} N_1(x_1) & \cdots & N_n(x_1) \\ \vdots & \ddots & \vdots \\ N_1(x_n) & \cdots & N_n(x_n) \end{pmatrix}_{n \times n}$

$$\int_{-\infty}^{+\infty} \left(\sum_{j=1}^{n} \beta_j N_j''(x) \right)^2 dx = \int_{-\infty}^{+\infty} \left(\sum_{l=1}^{n} \beta_l N_l''(x) \right) \left(\sum_{m=1}^{n} \beta_m N_m''(x) \right) dx$$

$$= \int_{-\infty}^{+\infty} \sum_{l=1}^{n} \sum_{m=1}^{n} \beta_l \beta_m N_l''(x) N_m''(x) \, dx$$

$$\overset{?}{=} \sum_{l=1}^{n} \sum_{m=1}^{n} \beta_m \beta_l \int_{-\infty}^{+\infty} N_l''(x) N_m''(x) \, dx$$

$$= \underset{\sim}{\beta}^T \Omega_N \underset{\sim}{\beta} \quad \text{where} \quad \underset{\sim}{\beta} = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix} \quad \& \quad \Omega_N = [\omega_{ij}]$$

an $n \times n$ matrix with

$$\omega_{ij} = \int_{-\infty}^{+\infty} N_i''(x) N_j''(x) dx$$

Hence, the objective function is

$$S(\underset{\sim}{\beta}) = (\underset{\sim}{y} - N\underset{\sim}{\beta})^T (\underset{\sim}{y} - N\underset{\sim}{\beta}) + \lambda \underset{\sim}{\beta}^T \Omega_N \underset{\sim}{\beta}$$

$$= \underset{\sim}{y}^T \underset{\sim}{y} - 2\underset{\sim}{y}^T N \underset{\sim}{\beta} + \underset{\sim}{\beta}^T (N^T N + \lambda \Omega_N) \underset{\sim}{\beta}$$

$$\frac{\partial S(\underset{\sim}{\beta})}{\partial \underset{\sim}{\beta}} = 0 \Rightarrow -2\underset{\sim}{y}^T N + 2\hat{\underset{\sim}{\beta}}^T (N^T N + \lambda \Omega_N)^T = 0$$

$$\Rightarrow \boxed{\hat{\underset{\sim}{\beta}} = (N^T N + \lambda \Omega_N)^{-1} N^T \underset{\sim}{y}}$$

$\hat{\beta}_\lambda$: { Linear Function of $\underset{\sim}{y}$
{ Similar to the Solution of Ridge

Choosing λ

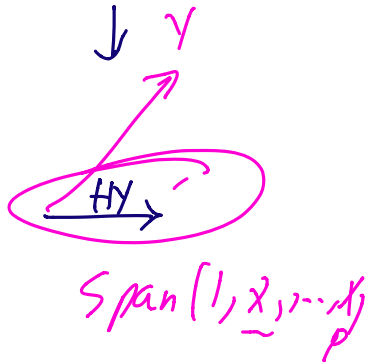
In smoothing splines there is a knot at every point x_i, $i = 1, .., n$, so we don't need to worry about choosing the number and/or location of the knots, but we have to choose λ.

The tuning parameter (penalty factor) λ can be chosen using cross-validation

We will show, through connecting λ to a measure of complexity (or roughness) of the fitted model, that the LOOCV can be simplified to tune λ in a computationally efficient way.

Degrees of Freedom

- Recall multiple linear regression $\mathbf{Y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$. The degrees of freedom (number of parameters) of this model is $p+1$: # of explanatory variables + 1 (+1 for β_0).



- Recall that

$$\widehat{\mathbf{y}} = X\widehat{\boldsymbol{\beta}} = X(X^T X)^{-1} X^T \mathbf{y} = H\mathbf{y}$$

where $\text{trace}(H) = p+1$ => $\text{trace}(H) = df$

- The *hat-matrix* H is a projection matrix, i.e. it is idempotent $H^2 = H$.

$H\hat{Y} = \hat{Y}$

- Observation: $\text{trace}(H) = Rank(X)$ is the dimension of the *projection space*.

(Effective) Degrees of Freedom Method

We have

$$\begin{aligned}\widehat{\boldsymbol{\mu}} &= \left(\widehat{\mu}(x_1), \widehat{\mu}(x_2), \ldots \widehat{\mu}(x_n)\right)^T \\ &= N\widehat{\boldsymbol{\beta}} \\ &= N\left(N^T N + \lambda \boldsymbol{\Omega}_N\right)^{-1} N^T \mathbf{y} \\ &= S_\lambda \mathbf{y}\end{aligned}$$

See the book Elements of Statistical Learning for details.

$S_\lambda \times S_\lambda$ + positive definite matrix $= S_\lambda$

$\Rightarrow S_\lambda \times S_\lambda \neq S_\lambda$ in fact $S_\lambda \times S_\lambda \prec S_\lambda$

hence a smoothing spline is a *linear smoother.*

S_λ is not a projection matrix (why?), hence we cannot say that the dimension of the projection spate (df) is trace(S_λ).

However, by analogy, we can define the *effective degrees of freedom* of a smoothing spline to be

$$df_\lambda = \text{trace}(S_\lambda) = \sum_{i=1}^{n} \{S_\lambda\}_{ii}$$

Leave-One-Out and Generalized Cross-Validation

There is a monotone decreasing relationship between λ and df_λ. This means cross-validation can be done on either λ or on df_λ.

- **LOOCV:**

Mean Square of Residual

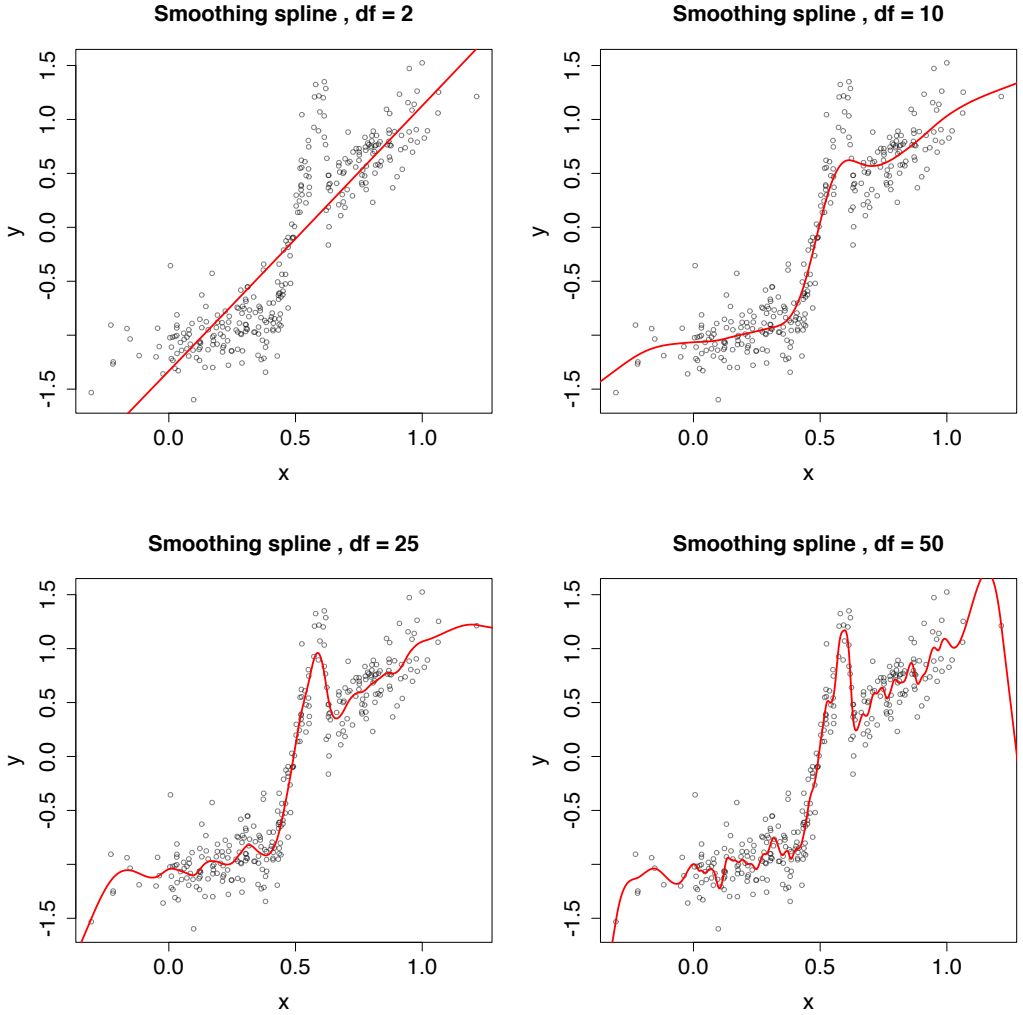
$$RSS_{CV}(\lambda) = \sum_{i=1}^{n} \left(y_i - \widehat{\mu}_\lambda^{(-i)}(x_i) \right)^2 = \sum_{i=1}^{n} \left(\frac{y_i - \widehat{\mu}_\lambda(x_i)}{1 - \{S_\lambda\}_{ii}} \right)^2$$

Some references use MSR instead (not R though)

- **Generalized Cross-validation:**

$$RSS_{GCV}(\lambda) = \sum_{i=1}^{n} \left(\frac{y_i - \widehat{\mu}_\lambda(x_i)}{1 - \frac{1}{n}\text{trace}(S_\lambda)} \right)^2$$

Example : Simulated Data



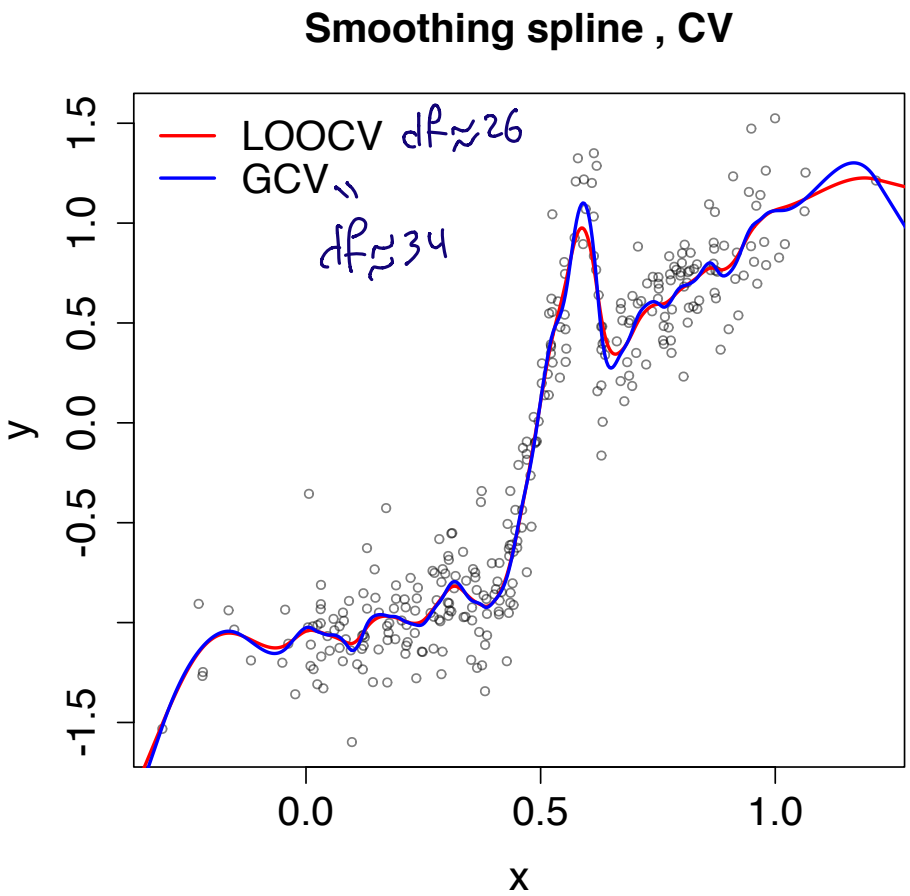
LOOCV : Simulated Data

```
> sm.LOOCV <- smooth.spline(x, y, cv = TRUE)
> sm.LOOCV
Call:
smooth.spline(x = x, y = y, cv = TRUE)

Smoothing Parameter  spar= 0.6402688  lambda= 7.07313e-06
                                                (14 iterations)
Equivalent Degrees of Freedom (Df): 25.82318
Penalized Criterion (RSS): 12.62976
PRESS(l.o.o. CV): 54.50422

> sm.LOOCV$lambda
[1] 7.07313e-06
> sm.LOOCV$df
[1] 25.82318
```

GCV : Simulated Data

```
> sm.GCV <- smooth.spline(x, y, cv = FALSE)
> sm.GCV
Call:
smooth.spline(x = x, y = y, cv = FALSE)

Smoothing Parameter  spar= 0.5603612  lambda= 1.87198e-06
                                      (11 iterations)
Equivalent Degrees of Freedom (Df): 33.9024
Penalized Criterion (RSS): 11.7393
GCV: 1.65062

> sm.GCV$lambda
[1] 1.87198e-06
> sm.GCV$df
[1] 33.9024
```

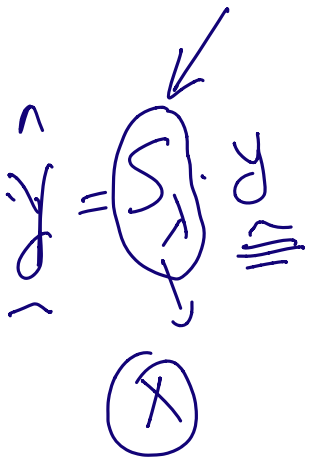
Cross-validated Fits : Simulated Data



Note

Note: In the interest of time, the material from here up to slide 103 can be skipped.

This material discuss the geometric and linear algebraic details of smoothing splines.



Eigen Decomposition

Consider the square matrix $A_{p\times p}$ with eigenvalues $\lambda_1, ..., \lambda_p$ and corresponding eigenvectors $\mathbf{u}_1$,...,$\mathbf{u}_p$. Let $U = [\mathbf{u}_1 \ , \ \mathbf{u}_2 \ , \ ... \ , \ \mathbf{u}_p]$ and $\Lambda = \text{diag}(\lambda_1, ..., \lambda_p)$. Then A can be written as

$$A = U\Lambda U^{-1} = \sum_{i=1}^{p} \lambda_i \, \mathbf{u}_i \mathbf{u}_i^* \rightarrow \text{eigen decomposition}$$

Properties:

- If A is symmetric, then the eigenvectors $\mathbf{u}_1$,...,$\mathbf{u}_p$ are orthogonal.
- $A^2 = U\Lambda^2 U^{-1}$
- If A is a (real) symmetric matrix, then
  - $U^{-1} = U^T$, hence $A = U\Lambda U^T \rightarrow A = \sum_{i=1}^{p} \lambda_i \, \mathbf{u}_i \mathbf{u}_i^T$

Singular Value Decomposition

If

- A is not a square matrix, or
- U is not an invertible matrix, e.g. $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, then U^{-1} does not exist

then A does not have an eigen decomposition. For such cases, we use the *singular value decomposition* of A.

Consider the matrix $A_{n \times p}$ where $(n \geq p)$. The *singular value decomposition* (SVD) of A is the product of three matrices

$$A_{n \times p} = U_{n \times n} \, \Sigma_{n \times p} \, V^T_{p \times p} = \sum_{i=1}^{p} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

$U U^T = I_n$ $\quad$ $V V^T = I_p$

where U and V have orthonormal columns and $\Sigma = \begin{pmatrix} \text{diag}(\sigma_1, \ldots, \sigma_p) \\ \mathbf{0}_{(n-p) \times p} \end{pmatrix}$.

If A is a symmetric (positive semidefinite) matrix, then SVD and engen decomposition of A are the same thing.

Regression : An Eigen Decomposition Approach

- In the regression model $\mathbf{Y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$, the hat matrix $H = X(X^TX)^{-1}X^T$ is the projection matrix.

- The n-dimensional fitted mean is $\widehat{\boldsymbol{\mu}} = H\mathbf{y}$: the orthogonal projection of $\mathbf{y}$ on to the space spanned by the columns of X, i.e. $colsp(X)$.

- Assuming $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_n \geq 0$ are the eigen-values of H with corresponding eigen-vactors $\mathbf{u}_1, ..., \mathbf{u}_n$, we have

$$H = \sum_{i=1}^{n} \rho_i \mathbf{u}_i \mathbf{u}_i^T \implies \widehat{\boldsymbol{\mu}} = \sum_{i=1}^{n} \rho_i \, \mathbf{u}_i \, \mathbf{u}_i^T \, \mathbf{y} = \sum_{i=1}^{n} \mathbf{u}_i \rho_i < \mathbf{u}_i \, , \, \mathbf{y} >$$

- Since H is a projection matrix (idempotent), the eigenvalues of H are 0 or 1. It can be shown that the first r eigenvalues are 1 (the rest are 0) where $r = rank(X)$.

Regression : An Eigen Decomposition Approach

- The general case of cubic splines is the same as regression.
- Replace the design matrix X with the basis matrix B and the corresponding projection matrix is $H_B = B(B^T B)^{-1} B^T$.
- When using `bs(...)` or `ns(...)` in `R`, we could specify the degrees of freedom `df` and have the function choose the appropriate number (and location) of the knots with which to build B.
- Can we parameterize a smoothing spline in a similar way?

Smoothing Splines : An Eigen Decomposition Approach

$\hat{\mu} = S_\lambda \underset{\sim}{y}$ where $S_\lambda = N(N^T N + \lambda \Omega_N)^{-1} N^T$

Note that $S_\lambda = N(N^T N + \lambda \Omega_N)^{-1} N^T$

inverse of N^T $\leftarrow = [N^{-T}(N^T N + \lambda \Omega_N) N^{-1}]^{-1}$

$= [I_n + \lambda N^{-T} \Omega_N N^{-1}]^{-1}$

$= (I_n + \lambda K)^{-1}$ where $K = N^{-T} \Omega_N N^{-1}$

1) does not involve penalty λ

2) Known as penalty matrix.

$\Rightarrow S_\lambda = (I_n + \lambda K)^{-1}$

we'll use this result for eigen decomposition of S_λ.

Smoothing Splines : An Eigen Decomposition Approach

Assuming the form $\mu(x) = N\boldsymbol{\beta}$, where N is the $n \times n$ matrix of basis functions calculated at $x_1, ..., x_n$, we have

$$\begin{aligned} RSS(\mu\ ,\ \lambda) &= \sum_{i=1}^{n} (y_i - \mu(x_i))^2 + \lambda \int \left(\mu''(t)\right)^2 dt \\ &= (\mathbf{y} - N\boldsymbol{\beta})^T (\mathbf{y} - N\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^T \Omega_N \boldsymbol{\beta} \\ &= (\mathbf{y} - N\boldsymbol{\beta})^T (\mathbf{y} - N\boldsymbol{\beta}) + \lambda \boldsymbol{\mu}^T K \boldsymbol{\mu} \end{aligned}$$

where $\boldsymbol{\mu}^T = \big(\mu(x_1), ..., \mu(x_n)\big)$.

The fitted model/values is/are

$$\begin{aligned} \widehat{\boldsymbol{\mu}} &= S_\lambda \mathbf{y} \\ &= (I_n + \lambda K)^{-1} \mathbf{y} \end{aligned}$$

eigen values of S_λ which depend on λ.

We can write $\hat{\underset{\sim}{\mu}} = \sum_{i=1}^{n} \rho_i(\lambda)\, \underset{\sim}{u}_i \underset{\sim}{u}_i^T \underset{\sim}{y}$ where $\rho_i(\lambda)$ are the $i = 1, \ldots, n$

Note that K is a real matrix & symmetric. Hence, we write it as $K = V \times D \times V^T$

$$\downarrow$$

$$\text{diag}(d_1, \ldots, d_n) \quad d_1 \geq d_2 \geq \cdots \geq d_n$$

However, $S_\lambda = (I_n + \lambda K)^{-1} = (I_n + \lambda V D V^T)^{-1}$

$$\overset{?}{=} V(I_n + \lambda D)^{-1} V^T$$

$$\Rightarrow \ell_i(\lambda) = \frac{1}{1 + \lambda d_{n-i+1}}$$

$i = 1, 2, \ldots, n$

eigenvalues of S_λ.

Note: $d_n = d_{n-1} = 0$ (why?)

corresponding to $\hat{\nu}_1(x) = a$

$\hat{\nu}_2(x) = a^* + b^* x$

$$\Rightarrow \hat{\mu} = \sum_{i=1}^{n} \ell_i(\lambda) \underset{\sim}{v_i} \underset{\sim}{v_i}^T \underset{\sim}{y}$$

$$= \sum_{i=1}^{n} \ell_i(\lambda) \underset{\sim}{v_i} \langle \underset{\sim}{v_i}, \underset{\sim}{y} \rangle = \sum_{i=1}^{n} \underset{\sim}{v_i} \frac{1}{1 + \lambda d_{n-i+1}} \langle \underset{\sim}{v_i}, \underset{\sim}{y} \rangle$$

Smoothing Splines : An Eigen Decomposition Approach

- The difference btween this and p-order splines is that the eigen-values $\rho_i(\lambda)$ are not just zero or one. The two largest are 1 (corresponding to $d_1 = d_2 = 0$) and the rest are less than one.

Smoothness penalty (2^{nd} derivative) is 0 for constant & linear fits

or better say roughness

- Since $\mathbf{H}_B$ is a projection matrix, (potentially) r directions are selected and the rest are dropped. That is the nature of a projection operator and so this kind of spline is sometimes called a **projection smoother**.

- In contrast, $S_\lambda \times S_\lambda \preceq S_\lambda$ (not idempotent, in fact, the right-hand-side exceeds the left-hand-side by a positive-definite matrix). $S_\lambda \times S_\lambda$ has smaller eigen-values than the original S_λ. Because of this shrinkage, the smoothing spline is sometimes called a **shrinking smoother.**

eigen values : $\rho_i = \frac{1}{1+\lambda d_i}$ not always 0 or 1 $\Rightarrow \rho_i^2 < \rho_i \Rightarrow$ Shrinkage

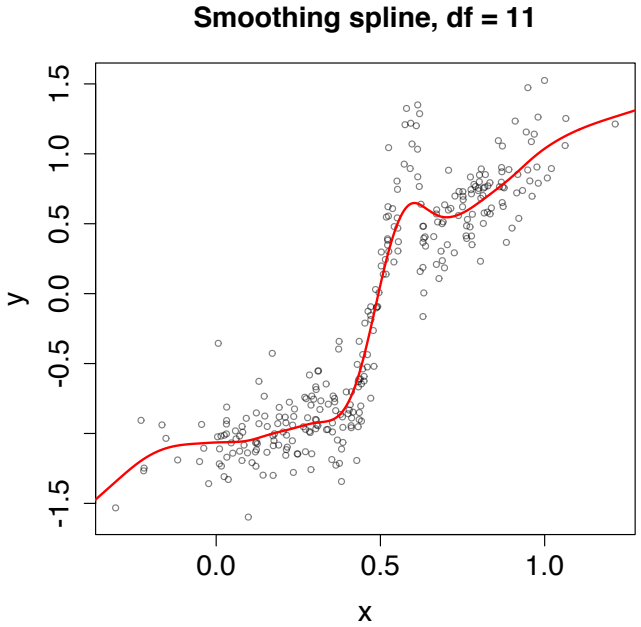
Effective Degrees of Freedom and λ

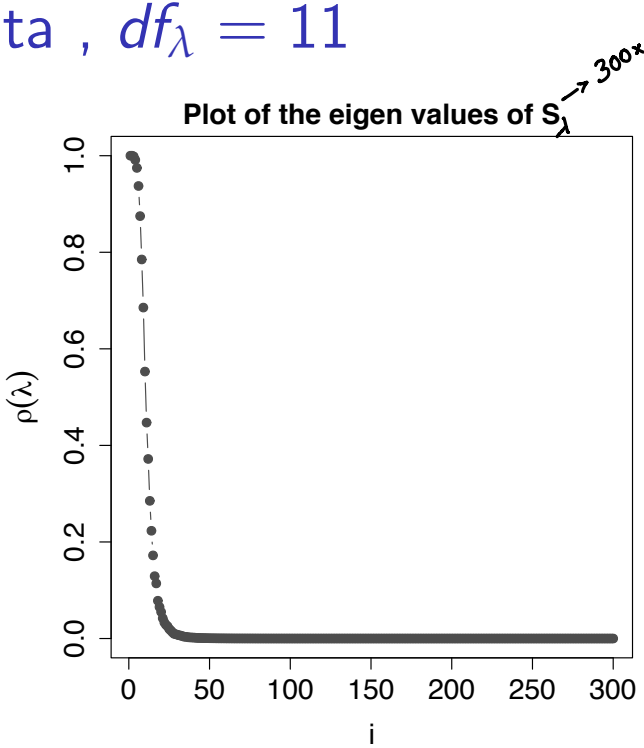
- The value $df_\lambda = \text{trace}(S_\lambda)$ is called the **effective degrees of freedom**. We have

$$df_\lambda = \sum_{i=1}^{n} \frac{1}{1+\lambda d_i}$$

  - df_λ is a decreasing function of λ
  - As $\lambda \to 0$: $df = 0$, $S_\lambda = I \Rightarrow \hat{\mu}(x_i) = y_i$ (perfect fit)
    - This means no projection
  - As $\lambda \to \infty$: $df = 2$, $S_\lambda \to H$ (simple linear regression with S_λ being the Hat matrix)
- To get some sense of what basis functions different values of λ correspond to, we could get the eigen decomposition of S_λ for our simulated data example.

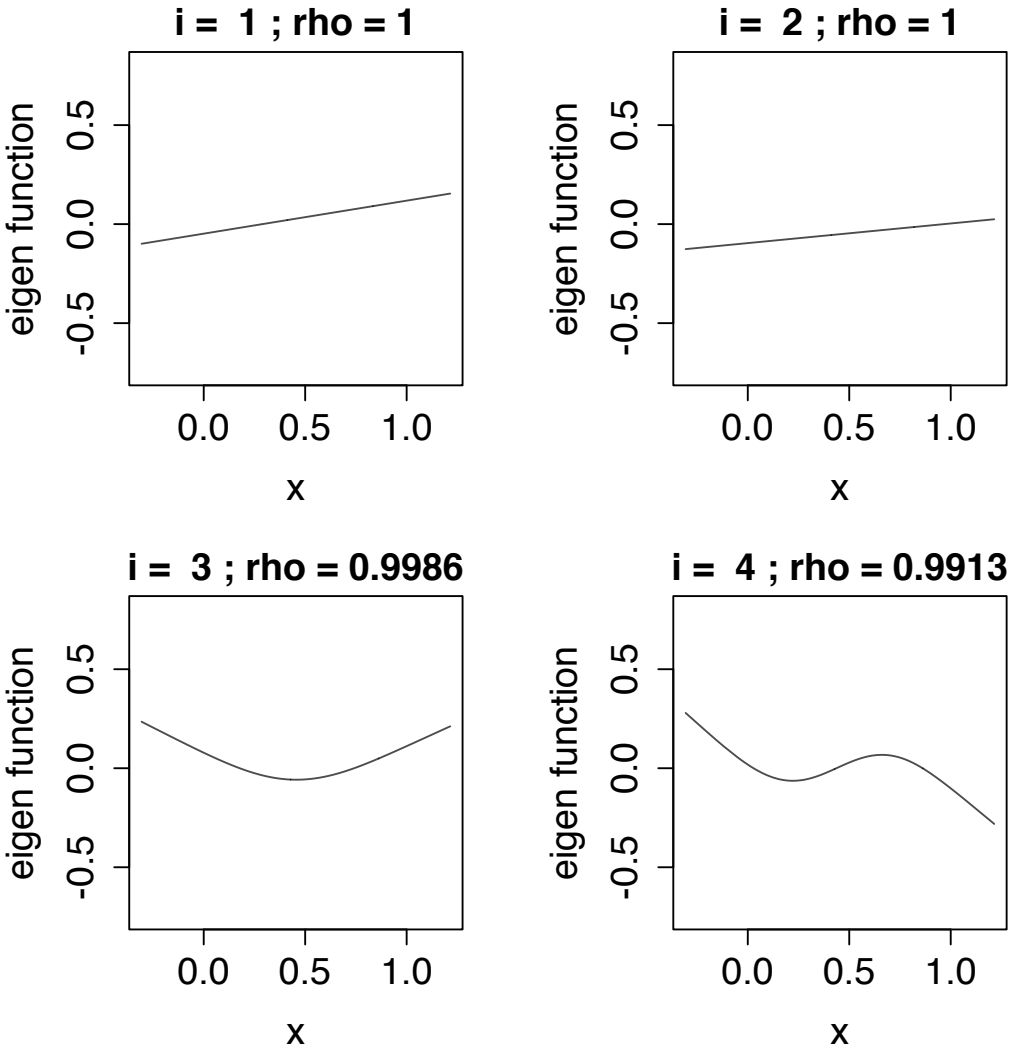
Simulated Data , $df_\lambda = 11$



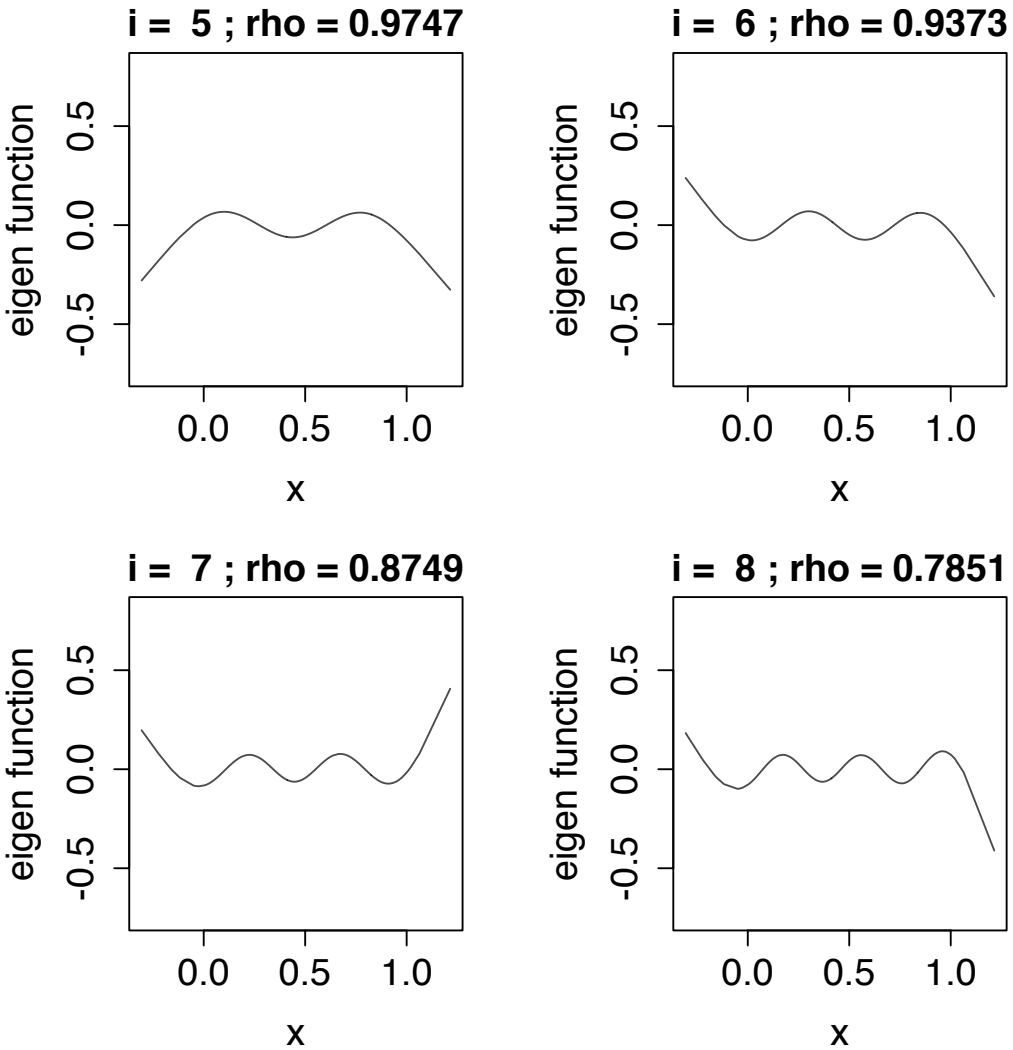


- $df_\lambda = 11$ since `sum(eigen(S)) = sum(diag(S)) =` 11.00184.
- $\rho_i(\lambda)$ drop off very quickly: components of **y** in the direction of the smallest eigenvalues are shrunk effectively to zero.
- What do the eigen-vectors corresponding to the largest eigen-values look like?

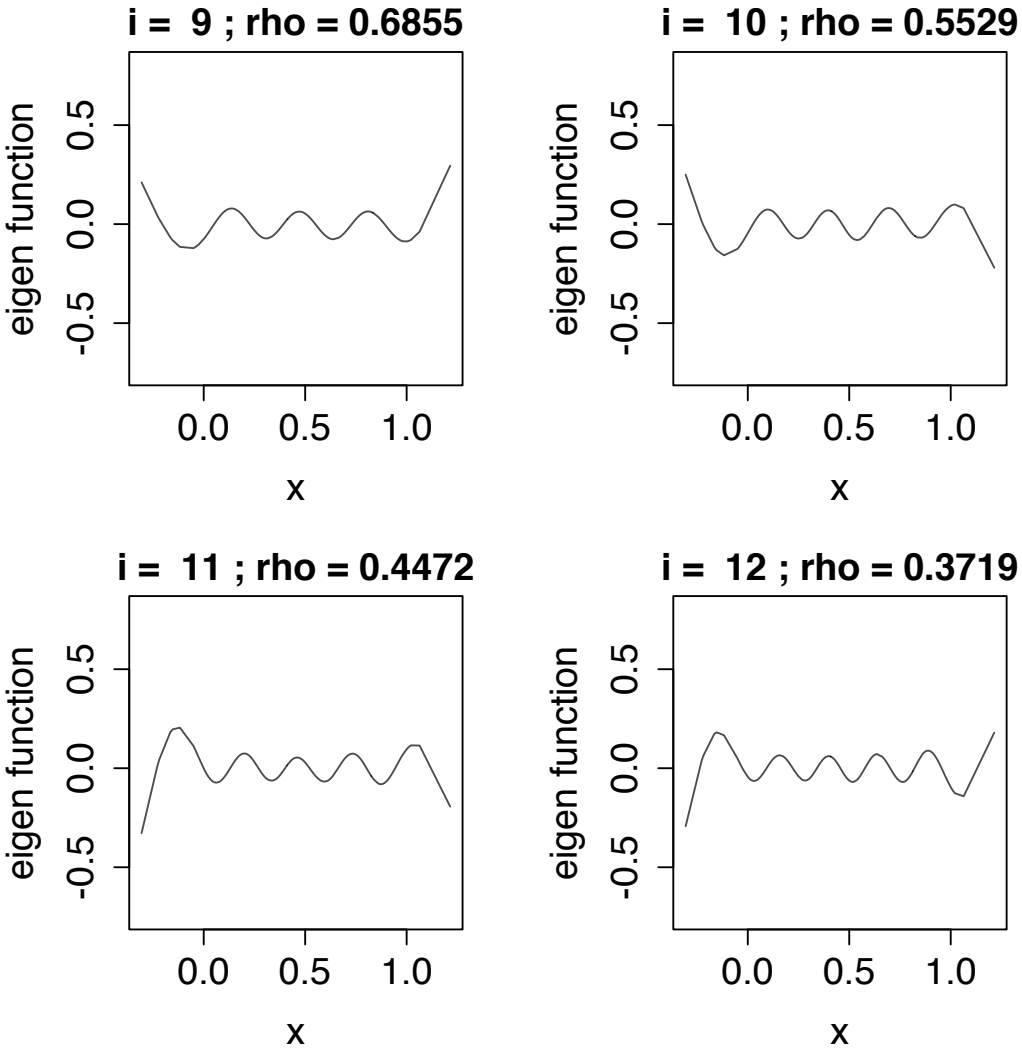
Eigen-vectors $\mathbf{u}_1, ..., \mathbf{u}_4$ of S_λ for $df_\lambda = 11$



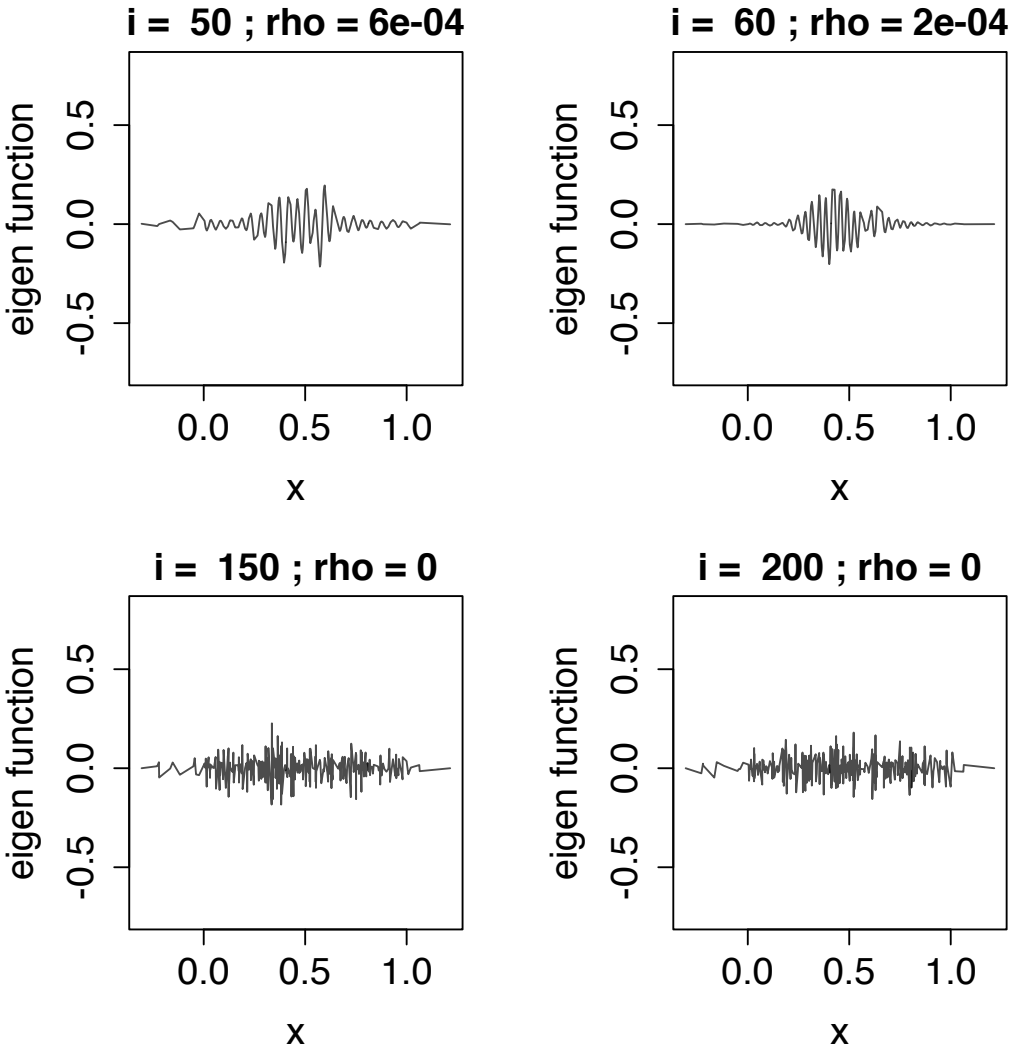
Eigen-vectors $\mathbf{u}_5, \ldots, \mathbf{u}_8$ of S_λ for $df_\lambda = 11$



Eigen-vectors $\mathbf{u}_9, \ldots, \mathbf{u}_{12}$ of S_λ for $df_\lambda = 11$



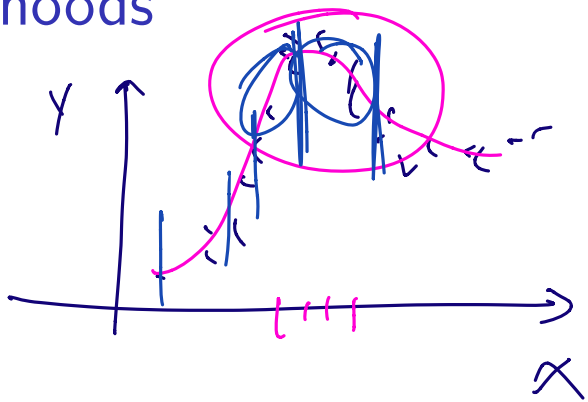
Farther out Eigen-vectors of S_λ for $df_\lambda = 11$



Agenda

- Introduction to Smoothing Methods
  - Polynomial Regression ✓
  - Basis Expansion ✓
- Splines
  - Cubic, Natural, and B-splines ✓
- Regularization Methods
  - Shrinkage Estimators: Ridge, LASSO, Elastic Net ✓
  - Smoothing Splines ✓
- Local Weighting
  - ~~Delta Neighbourhood and KNN~~
  - Kernel Smoothing
  - Local Smoothing and LOESS
- Multidimensional Smoothing (if time permits)
  - Tensor Product Bases
  - Thin-plate Splines
  - Additive Spline Model

Local Neighbourhoods



Regular splines: prespecified knots

Smoothing splines: fit the data more locally, i.e. knots at every unique point x_i

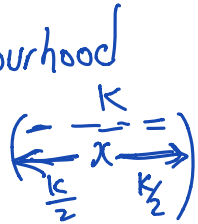
Alternative: Fit a model which focuses on each and every x locally, i.e. it fits to a "*neighbourhood*" of x.

K Nearest Neighbour Fitting

The simplest neighbourhood method is to fit a local average at each neighbourhood of x, i.e.

$$\widehat{\mu}(x) = Ave\big(y_i \mid x_i \in N_k(x)\big)$$

Neighbourhood



in which $N_k(x)$ is the set of k points nearest to x in squared distance.
(don't mix this up with N_d: constant distance neighbourhood)

To do this, we will find the k nearest neighbours of x for all values of x in the dataset.

The function `knn.reg` from the package `FNN` will compute this average for every point x_i in the data.

Let's apply this to our simulated dataset.

K Nearest Neighbour Fitting : Simulated Data

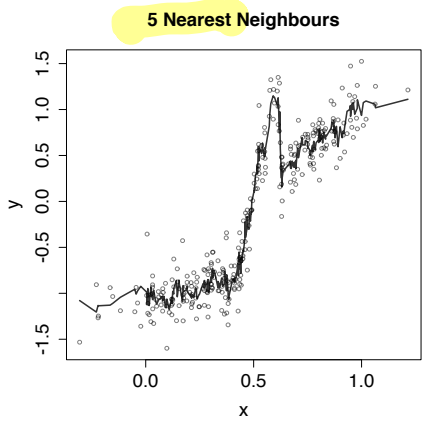
$n = 300$

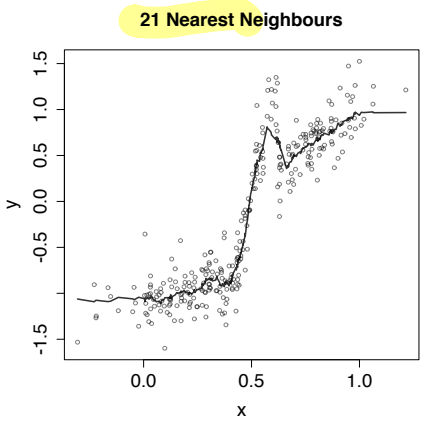
Press $= 16.22$
$R^2 = 0.921$

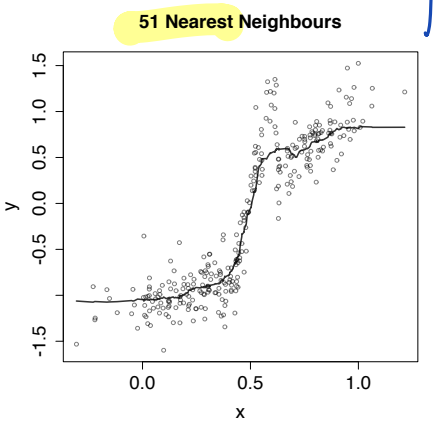
$\sum (y_i - \hat{y}_i)^2$

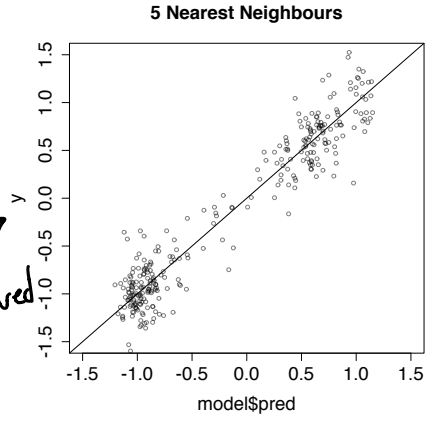
Press $= 15.77$
$R^2 = 0.923$

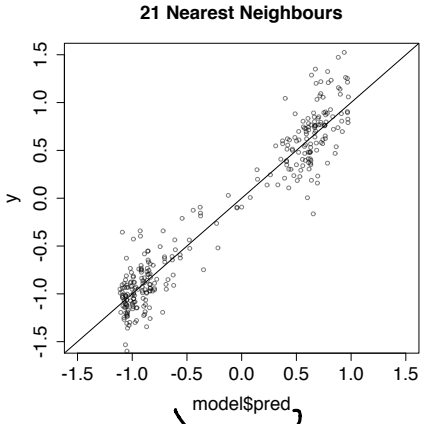
Press $= 19.45$
$R^2 = 0.906$

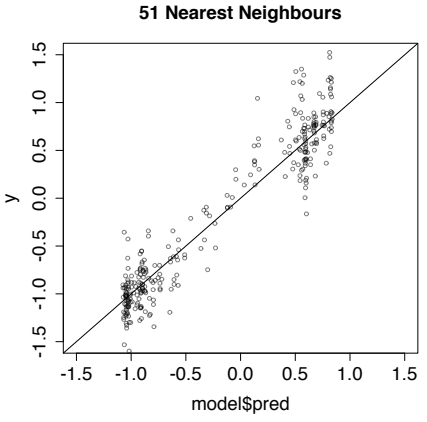












observed y

fitted values

Choosing K

- Small K:
  - complex/flexible model
  - high variability (wiggly)
  - low bias (potential for "chasing the noise")
- Large K:
  - inflexible model
  - low variability (smooth)
  - high bias

One strategy is to choose based on MSE/PRESS/R^2 (no test data), or we can do cross validation (with test data).

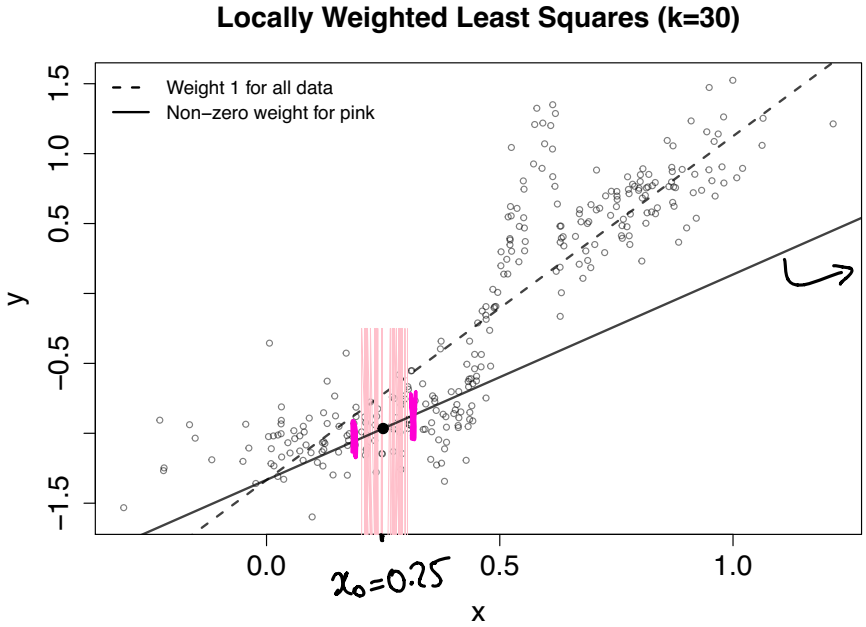
Increasing the Complexity of Local Fits

- As the size of local neighbourhoods increases, the smoother becomes the fitted function.
- We might also replace the averages with **any** fitted model based on the k nearest neighbours of any location x.

Local Weighting

You may think of local fitting as a model which uses all points, but those outside the local neighbourhood have zero weight (in the sense of weighted least squares).

Least-squares fit on the k nearest neighbours is a least-squares fit on all of the data but with weights that are 1 for points inside the neighbourhood and zero for points outside the neighbourhood.



local simple linear regression about $x_0 = 0.25$

$K = 30$

Local Weighting

The local least squares with zero-one weights can be written as

$$\begin{aligned}\widehat{\mu}(x) &= \arg\min_{\mu}\left[\sum_{i=1}^{N} w(x,x_i) r_i^2\right] \\ &= \arg\min_{\mu}\left[\sum_{i=1}^{N} w(x,x_i)(y_i-\mu(x_i))^2\right]\end{aligned}$$

where

$$w(x,x_i) = \begin{cases} 1 & x_i \in Nbhd(x) = N_k(x) \\ 0 & \text{otherwise.} \end{cases}$$

Local Weighting

Alternatively, we might not worry so much about the neighbourhood but rather choose the weights more judiciously.

Since we are trying to fit locally, we could choose higher weights for the closer points and lower weights for those farther away.

For example, we might consider weights that are proportional to a function, say $K(t)$ having the following properties:

- $\int K(t)dt = 1$
- $\int t\ K(t)dt = 0$
- $\int t^2\ K(t)dt < \infty$

The function $K(t)$ is called a *kernel function*.

Assuming also $K(t) \geq 0$, $\forall t$, then $K(t)$ could be a symmetric density function with mean 0 and finite variance.

Example of Kernel Functions

1) Epanechnikov kernel:

$$K(t) = \begin{cases} \frac{3}{4}\left(1 - t^2\right) & \text{for} \quad |t| < 1 \\ 0 & \text{otherwise.} \end{cases}$$

2) Tukey's tri-cube weight:

$\int_{-1}^{1} K(t)\,dt \neq 1$

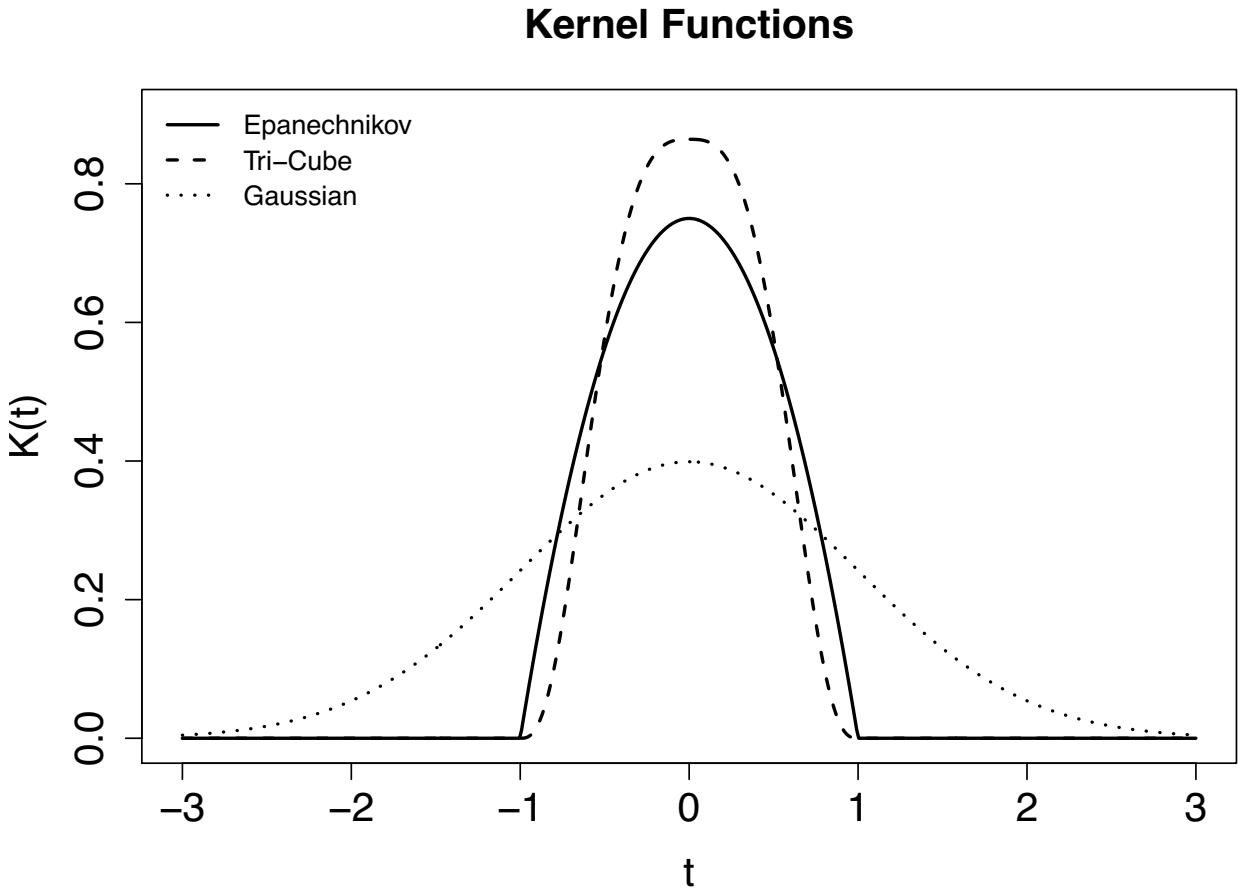
$$K(t) = \begin{cases} \left(1 - |t|^3\right)^3 & \text{for} \quad |t| \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

Normalizing factor $70/81$

3) Gaussian Kernel:

$$K(t) = \frac{1}{\sqrt{2\pi}}\, exp\left(-\frac{t^2}{2}\right).$$

Example of Kernel Functions



Kernel-weighted Average (Kernel Smoother)

Since we are interested in applying these functions locally, the kernel function $K(\cdot)$ is applied not to each x_i, but rather to the difference $x_i - x$.

Similarly, a means of controlling how quickly the weights diminish for any kernel is to introduce a scale parameter, say $h > 0$, called the *bandwidth*.

Putting these together, we evaluate the kernel function at $\frac{x_i - x}{h}$.

The smaller h, the more "local" the fit.

Kernel-weighted Average (Kernel Smoother)

Our weight function would be calculated as

$$w(x, x_i) = \frac{K\left(\frac{x_i - x}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{x_j - x}{h}\right)}$$

$$\begin{cases} 0 < \omega_i \leq 1 \\ \sum_i \omega_i = 1 \end{cases}$$

Assuming the squared error loss is used, and μ is modelled as a piecewise constant function, the previous piecewise constant estimate

$$\widehat{\mu}(x) = Ave\big(y_i \mid x_i \in N_k(x)\big)$$

is replaced with the *Nadaraya-Watson kernel-weighted average*:

$$\widehat{\mu}(x) = \sum_{i=1}^{n} w(x, x_i) y_i = \sum_{i=1}^{n} \left(\frac{K\left(\frac{x_i - x}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{x_j - x}{h}\right)} \right) y_i$$

Proof: Suppose the loss function is $\rho(r) = r^2$

$\hat{c}_x$ → piecewise constant.

$$\hat{\mu}(x) = \arg\min_{c_x} \left[\sum_{i=1}^{n} \omega(x, x_i)(y_i - c_x)^2 \right]$$

$$\min_c \left(\sum_{i=1}^{n} \omega(x, x_i)(y_i - c)^2 \right) = ? \rightarrow \frac{\partial}{\partial c} = 0$$

$$\Rightarrow -2 \sum_{i=1}^{n} \omega(x, x_i)(y_i - \hat{c}) = 0$$

$$\Rightarrow \sum_{i=1}^{n} \omega(x, x_i) y_i - \hat{c} \sum_{i=1}^{n} \omega(x, x_i) = 0$$

$$\Rightarrow \hat{c} = \sum_{i=1}^{n} \omega(x, x_i) \cdot y_i$$

$$\Rightarrow \hat{c} = \sum_{i=1}^{n} \left(\frac{K(\frac{x_i - x}{h})}{\sum_i K(\frac{x_i - x}{h})} \right) \cdot y_i \qquad \text{Q.E.D.}$$

Kernel Smoother : Local Linear Regression

Piecewise constant fit may be replaced by linear, quadratic, ... models.

The local fit at $x=x_0$: $\min_{\alpha(x_0),\beta(x_0)} \left[\sum_{i=1}^{n} \omega(x_0,x_i)\left(y_i-\alpha(x_0)-\beta(x_0)x_i\right)^2\right]$

which results in $\hat{\alpha}(x_0)$ & $\hat{\beta}(x_0)$.

$$\Rightarrow \hat{\mu}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)\cdot x$$

Define $b(x) = \begin{pmatrix} 1 \\ x \end{pmatrix} \Rightarrow b(x)^T = (1, x)$

$B_{n\times 2}$: regression matrix $\rightarrow$ i^{th} row of B is $b(x_i)^T$

$W_{n\times n}(x_0)$: $n\times n$ diagonal matrix $\rightarrow$ $W_{ii}(x_0) = \omega(x_0,x_i)$

$$\Rightarrow \hat{\mu}(x_0) = b(x_0)^T\hat{\beta} = \underbrace{b(x_0)^T\left[B^T W(x_0) B\right]^{-1} B^T W(x_0) y}_{\text{no } y \text{ here}}$$

$$= \sum_{i=1}^{n} l_i(x_0)\, y_i \rightarrow \text{linear Smoother}$$

Local Weights

Consider the weights

$$w(x, x_i) = \frac{K\left(\frac{x_i - x}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{x_j - x}{h}\right)}$$

where there are many choices for the kernel function K, some of which are Epanechnikov, Tukey's tri-cube , and Gaussian.

We can fit a weighted least squares model using these weights.

Note that the fitted model depends on the value of x. The idea is that the model is evaluated at many points x.

Illustration - Effect of Bandwidth h

Gaussian kernel

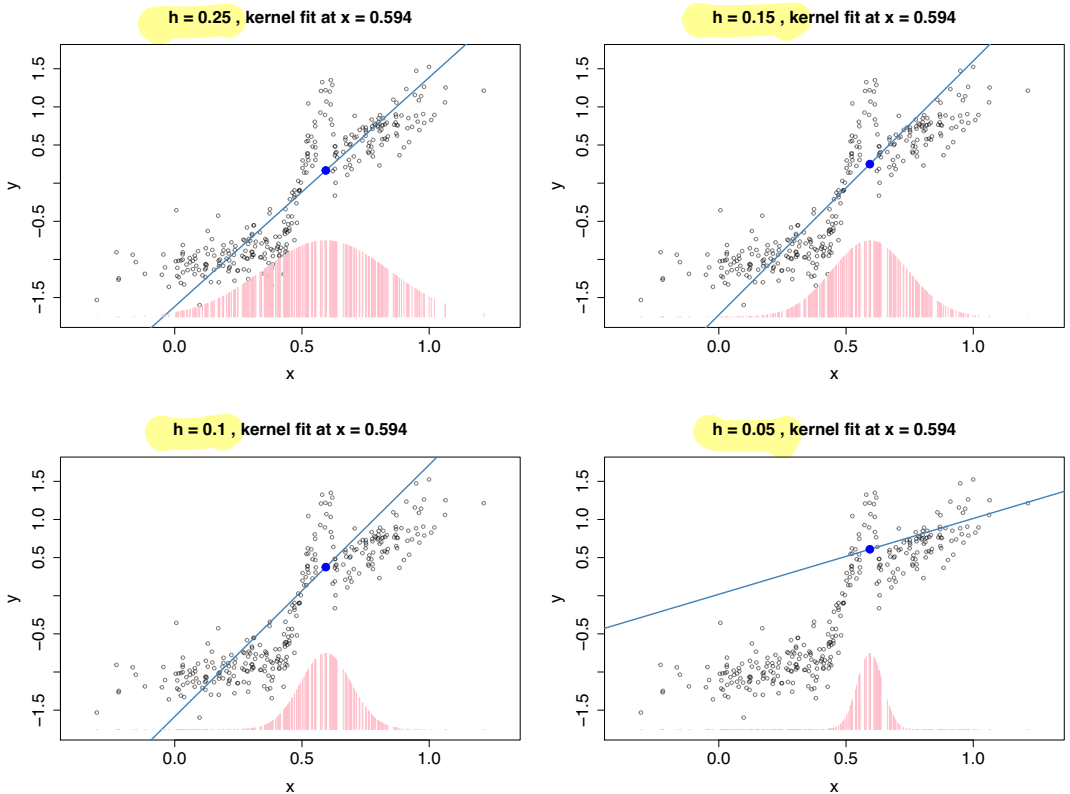
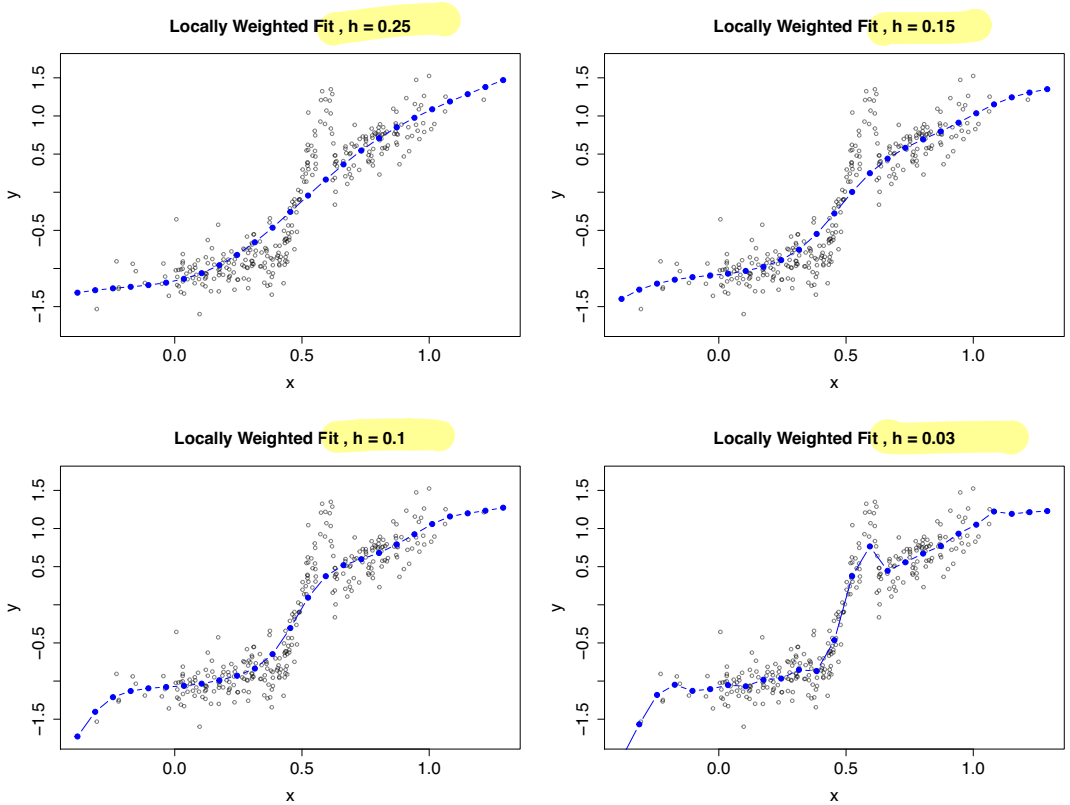
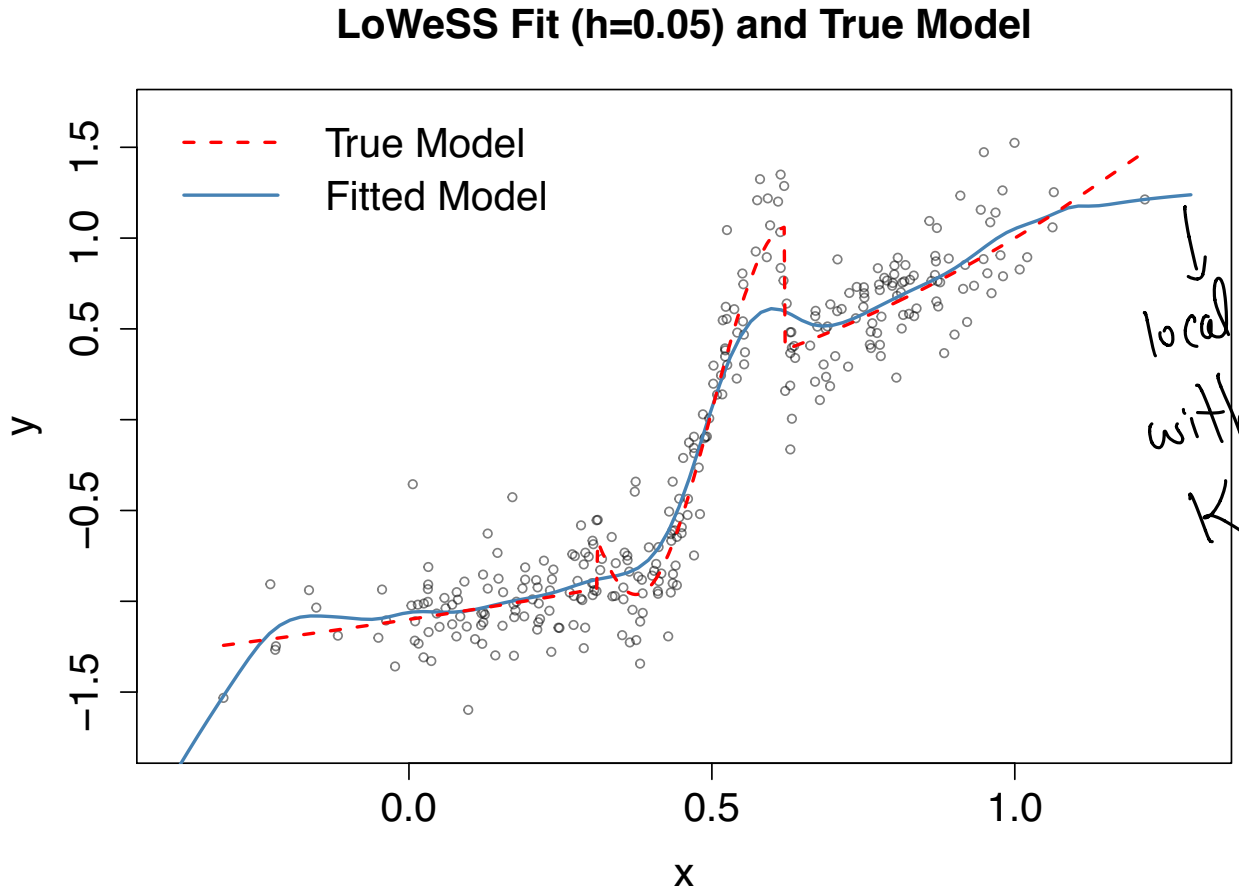


Illustration - Effect of Bandwidth h

- The smaller the bandwidth h, the more structures are captured, i.e. the more flexible is the model.
- Increase the number of points x to get a more precise plot of the fit.



Comparison to the True Model



What Remains?

The naive locally weighted sum of squares estimators have some difficulties which require attention:

- Choice of bandwidth h.
  - Also, we only looked at x locations but one might choose a proportion of the nearest x values.
- Choice of what weight function (kernel).
  - Perhaps a more robust choice that actually gives zero weight to points that are far away.
- What about the ends?
  - It seems that we can only estimate using data from one side of the endpoint. Should that affect the choice of bandwidth there? Or the weight function?

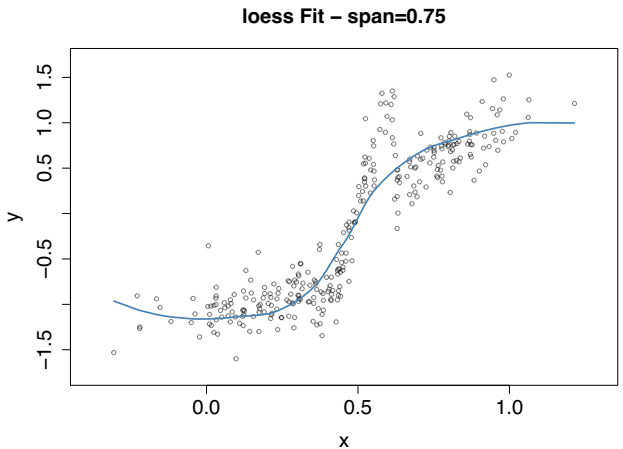
Using KNN and Local Weights Together

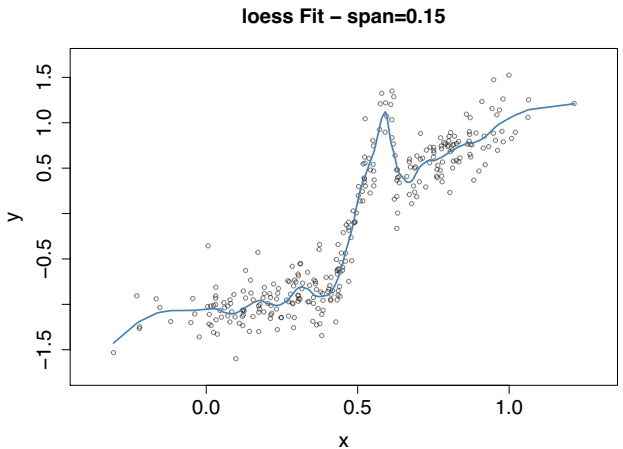
- The naive locally weighted sum of squares above did not define neighbourhoods, but rather used the scale parameter h to determine how weights would diminish.
- This means that where the data is densest in x, more points will appear in the estimation than where it is sparser.
- Let's impose the condition that only k points in a neighbourhood of x may have non-zero weights. all points outside of the neighbourhood will have zero weights.
- We will also use some kernel function to downweight points in the neighbourhood. The kernel will again be evaluated at $(x_i - x)/h$, but now we will choose h to be a function of the maximum distance $|x_i - x|$ over all points in the neighbourhood.

LOESS: Locally Estimated Scatterplot Smoothing

- We can use the function `loess` from the default-loaded `stats` package or the function `locpoly` from the `KernSmooth` package.

- Here, we focus on the function `loess`.





Notes on loess

The `loess` function uses a local neighbourhood determined by

- either its argument `span` (default=0.75) as the proportion of points, say α, taken as the local neighbourhood (roughly KNN with $k \approx n \times \alpha$)
- or its argument `enp.target`, this being the "number of equivalent parameters" in the model - like the effective model degrees of freedom, a measure of the complexity of the model. This too produces an equivalent proportion a of points in the local neighbourhood.

Notes on loess

Given the parameter $\alpha < 1$, the default weighting is given by Tukey's tri-cube weight function

$$K(t) = \begin{cases} (1 - |t|^3)^3 & \text{for} \quad |t| \leq 1 \\ 0 & \text{elsewhere} \end{cases}$$

For the ith point in the neighbourhood, we take

$$t_i = \frac{|x_i - x|}{\max_{j \in Nbhd(x)} |x_j - x|}.$$

If $\alpha > 1$, the above denominator is replaced by

$$(\alpha^{1/p}) \times \max_{j \in Nbhd(x)} |x_j - x|$$

where p is the number of explanatory variates in case there is more than 1. (in this case, $|x_i - x|$ is replaced everywhere by the Euclidean distance $\|\mathbf{x}_i - \mathbf{x}\|$)

Notes on loess

The function `loess` is not restricted to fitting local lines and can fit degree 0, 1, or 2 polynomial locally (in practice, typically degree 1 or 2 is used). Its default is 2.

The fitting mechanism is given by the parameter family and can be either gaussian (the default), which will use least-squares to fit the local polynomial, or symmetric which will begin with least squares and then perform a few iterations of an M-estimation using Tukey's bisquare weight function.

Some Models for $y = \mu(x) + \epsilon$

- Cubic Splines:

$$\mu(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^2 + \sum_{j=1}^{K} \beta_{j+3}(x - \xi_j)_+^3$$

- Natural cubic splines (solution to smoothing splines):

$$\widehat{\mu}(x) = \sum_{j=1}^{n} \beta_j N_j(x)$$

- Kernel regression (e.g. loess linear model):

$$\widehat{\mu}(x_0) = \widehat{\beta}_0(x_0) + \widehat{\beta}_1(x_0)x$$

where $\widehat{\boldsymbol{\beta}} = \left(\widehat{\beta}_0(x_0)\ ,\ \widehat{\beta}_1(x_0)\right)$ is computed from

$$\widehat{\boldsymbol{\beta}} = \arg \min_{\widehat{\beta}_0(x_0), \widehat{\beta}_1(x_0)} \left[\sum_{i=1}^{n} \left(\frac{K\left(\frac{x_i - x_0}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{x_j - x_0}{h}\right)} \right) \left(y_i - \beta_0(x_0) - \beta_1(x_0)x \right)^2 \right]$$

Comparing the Linear Smoothers

All of the above three methods are linear smoothers (why?) hence they have the same form. We should be able to look at them in much the same way– no matter how they were motivated or derived.

As an example, local regression smoothers like `loess` were built on kernel functions which gave higher weight to observations nearest the point $x = x_0$ of interest. We might expect, therefore, that in computing the fit at any given x that the coefficients multiplying any y would be higher for a y_i whose corresponding x_i was closer to x_0 than for one which was farther away. To check this, we might have a look at the coefficients of each y as a function of x.

How about other methods? Do they behave the same way?

Coefficient of y as a Function of x

A linear smoother has the form

$$\widehat{\mu}(x) = S\mathbf{y}$$

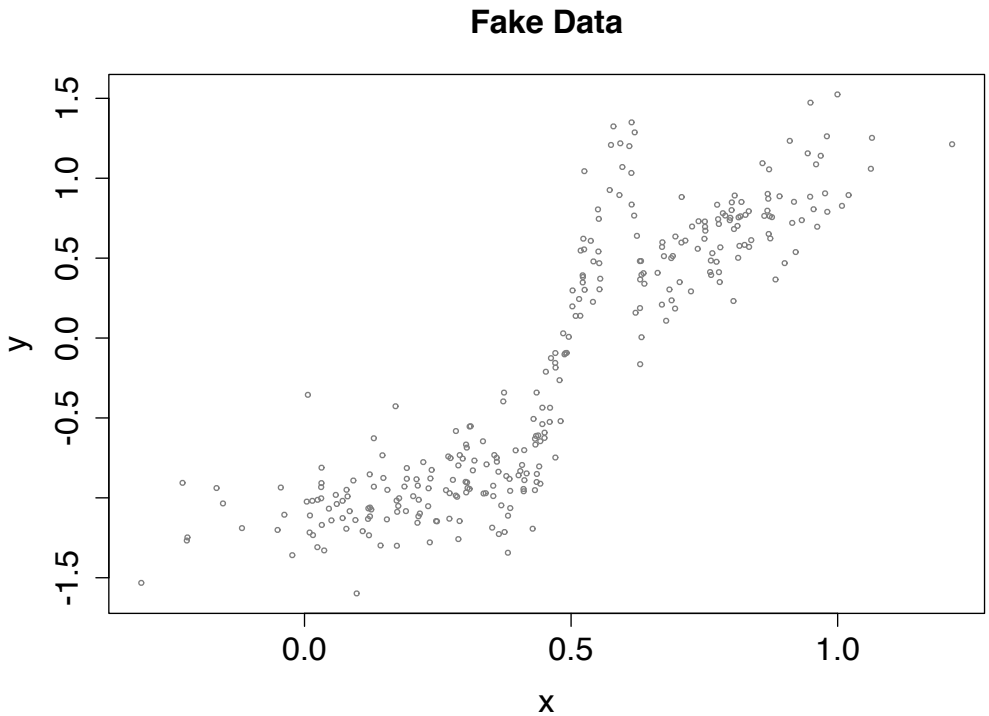
where, depending on the model (splines, kernel regression, etc.), the smoothing matrix S will change.

Note that common `R` packages/functions for the methods discussed so far do not output the smoothing matrix automatically, so you have to manually code it.

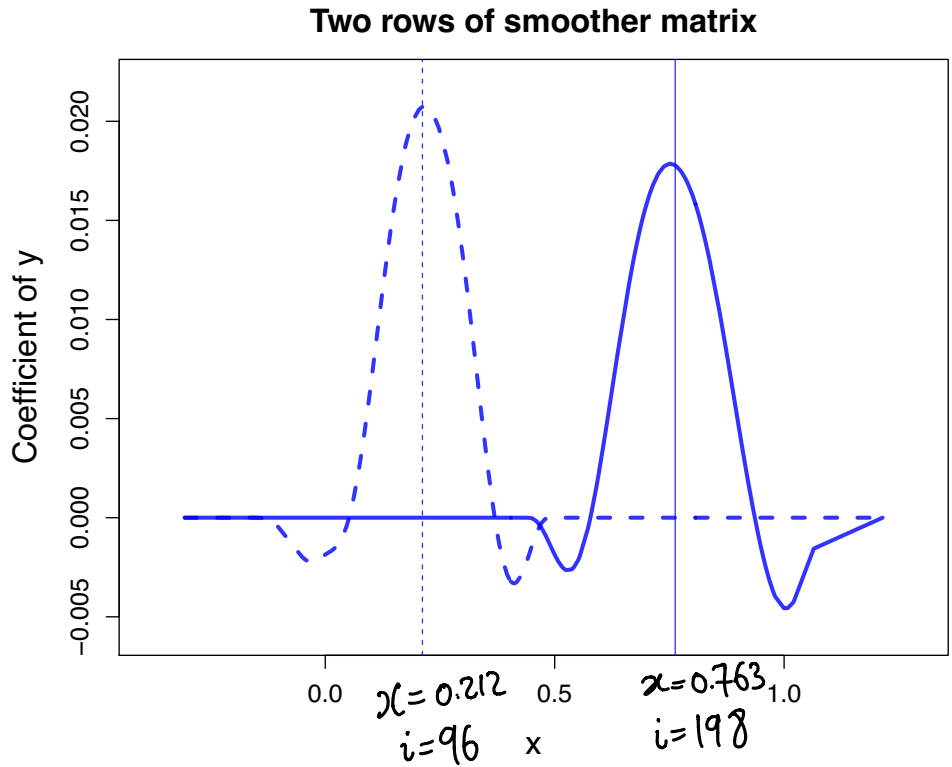
The i^{th} row of S is the coefficients of $\mathbf{y} = (y_1, ..., y_n)^T$ for $x = x_i$, i.e. $\widehat{\mu}(x_i) = S_i\mathbf{y}$ where S_i represents the i^{th} row of S.

Simulated Data

We will work with the following simulated data as a working example.

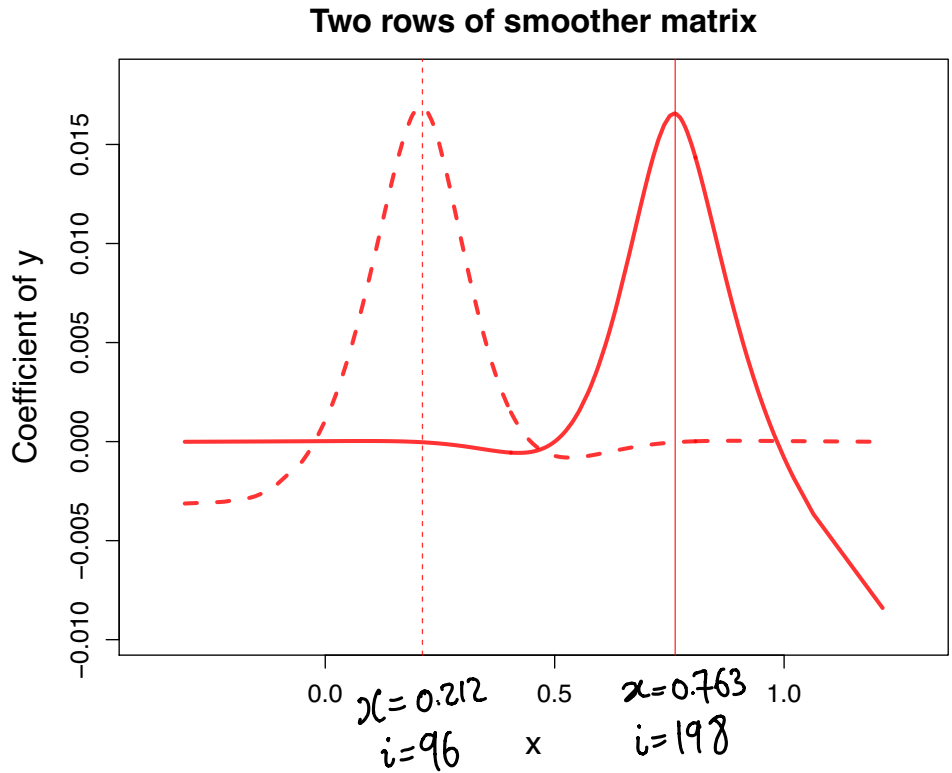


Coefficient of y as a Function of x For `loess`



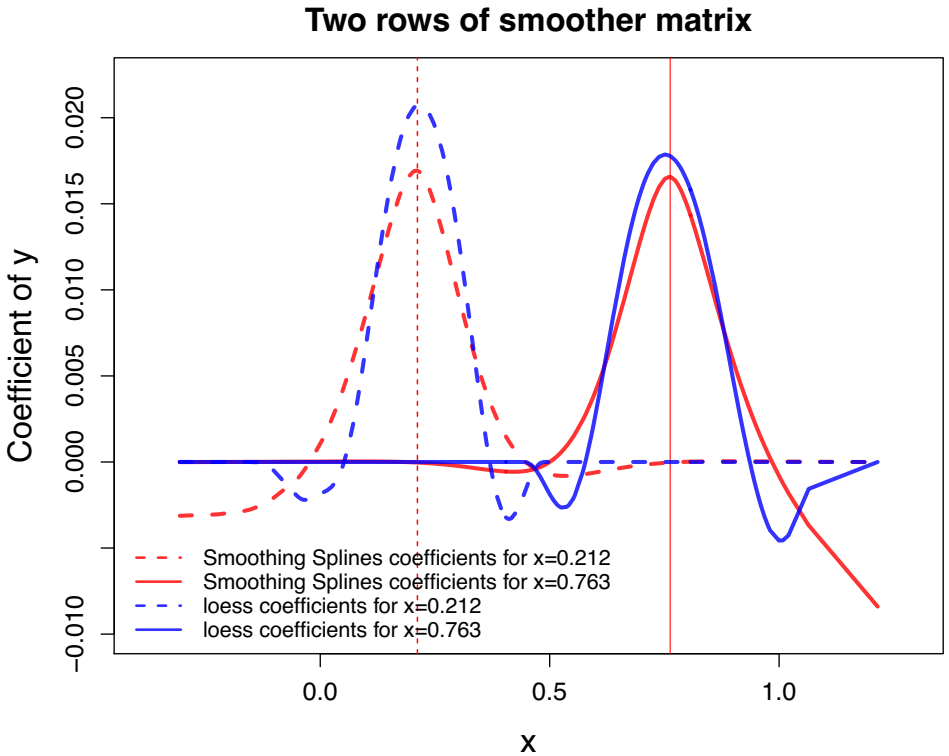
Note that the coefficients concentrate around the x value.

Coefficient of y as a Function of x For Smoothing Splines



As was the case with the local regression, the coefficient of y_i is higher the closer x value is to the value of x_i.

Coefficient of y as a Function of x



Decomposing the Smoothing Matrix

Recall the eigenvalue decomposition of the smoothing splines. We can do the same for any **linear smoother.**

In the general case of linear smoothing, the smoothing matrix may not be symmetric. As a result we will work with the *singular value decomposition* of the smoothing matrix S.

Decomposing the Smoothing Matrix

We decompose any smoother matrix S as

$$S = UD_\rho V^T$$

for $n \times n$ matrices $U = [\mathbf{U}_1, \ldots, \mathbf{U}_n]$, $V = [\mathbf{V}_1, \ldots, \mathbf{V}_n]$, and $D_\rho = diag(\rho_1, \ldots, \rho_n$ with $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_n \geq 0$ and

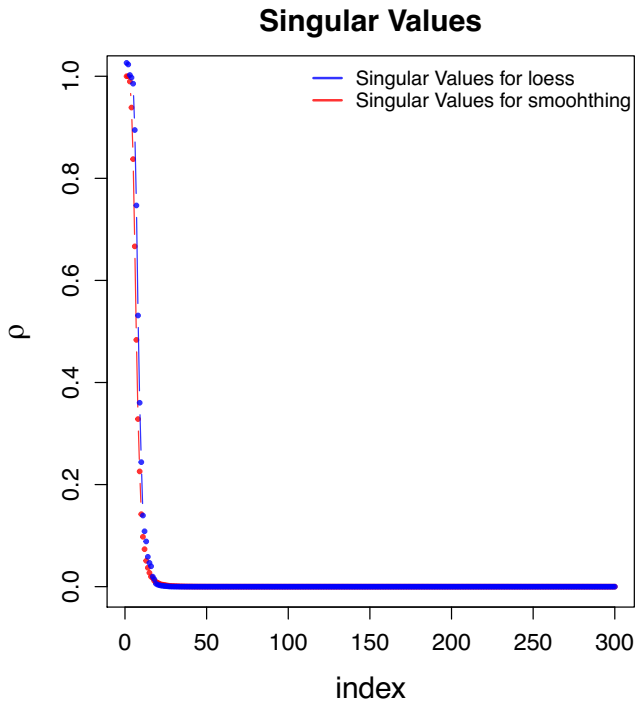
$$U^T U = I_n = V^T V.$$

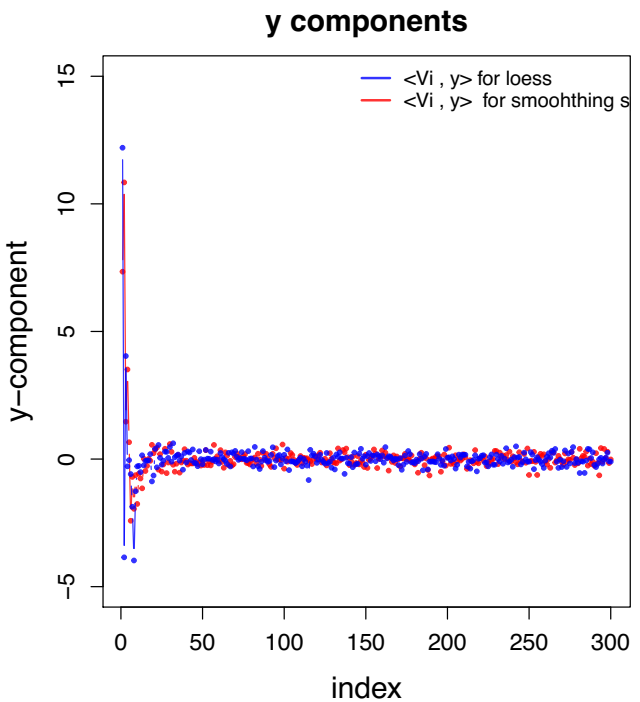
The smooth can now be written as

$$\begin{aligned} \widehat{\boldsymbol{\mu}} &= UD_\rho V^T \mathbf{y} \\ &= \sum_{i=1}^n \mathbf{U}_i \rho_i < \mathbf{V}_i, \mathbf{y} > \end{aligned}$$

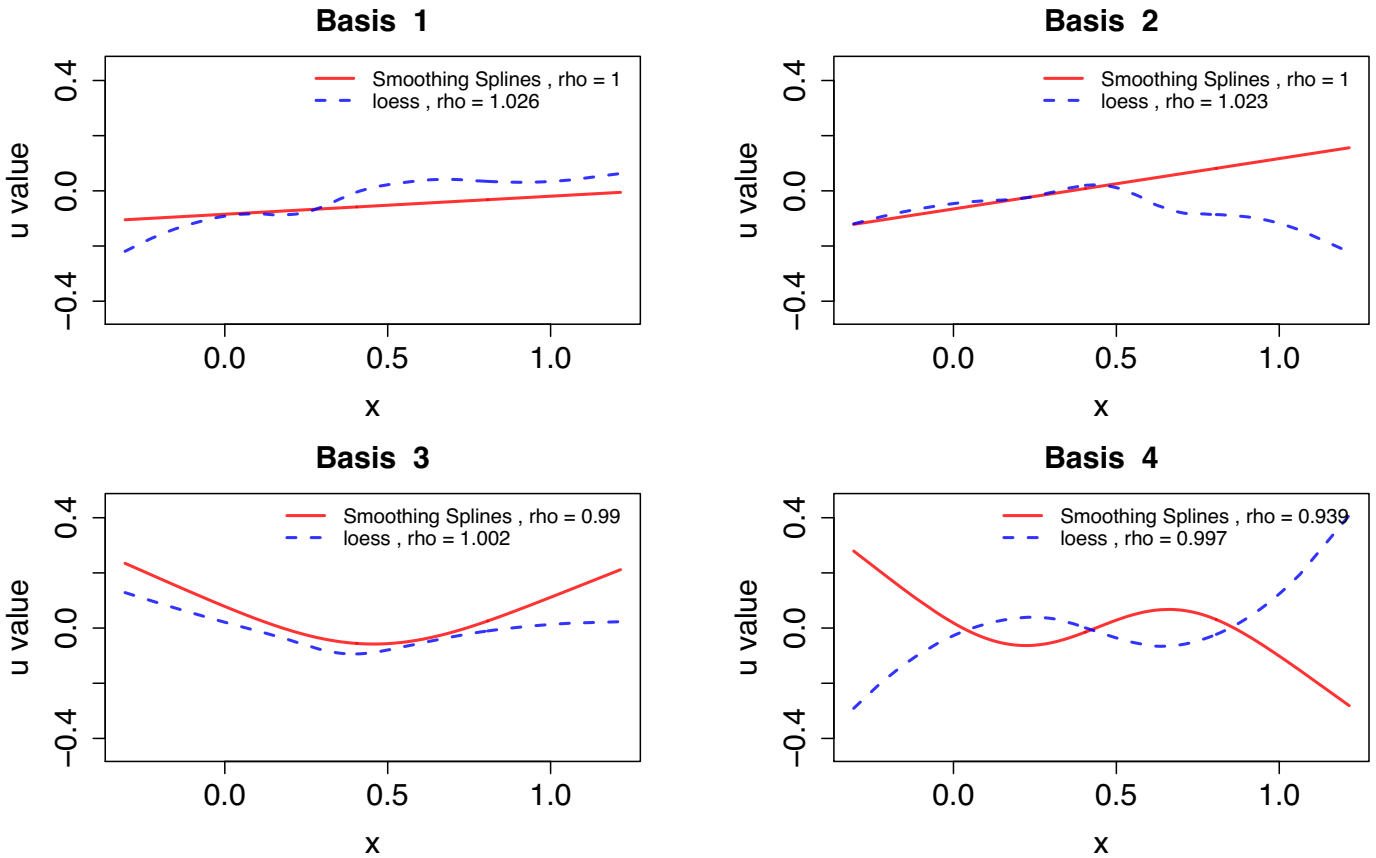
which separates into the basis vectors $\mathbf{U}_i$, the singular values ρ_i and the orthogonal component of $\mathbf{y}$ along the direction vectors $\mathbf{V}_i$.

Singular Values and **y** Components

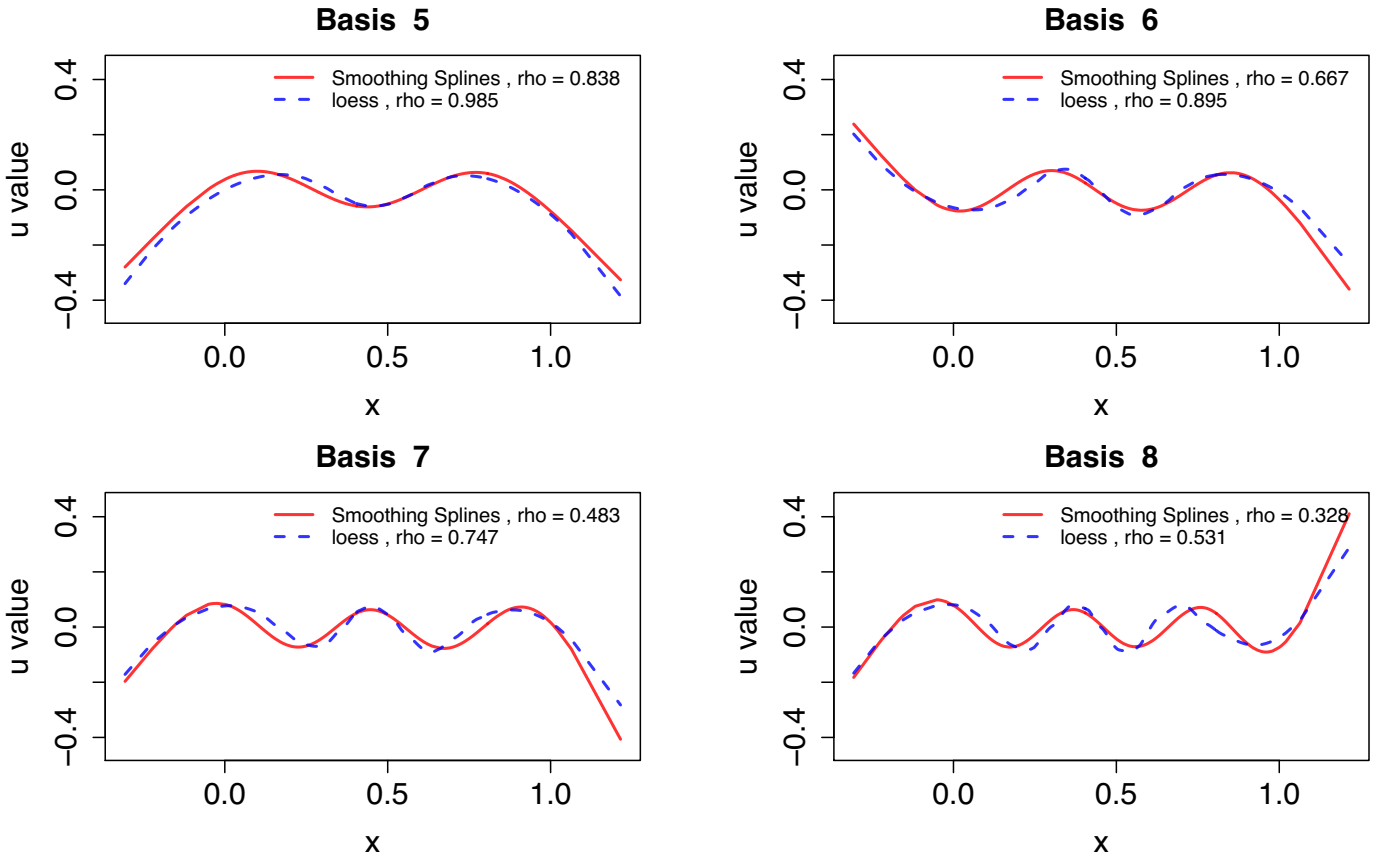




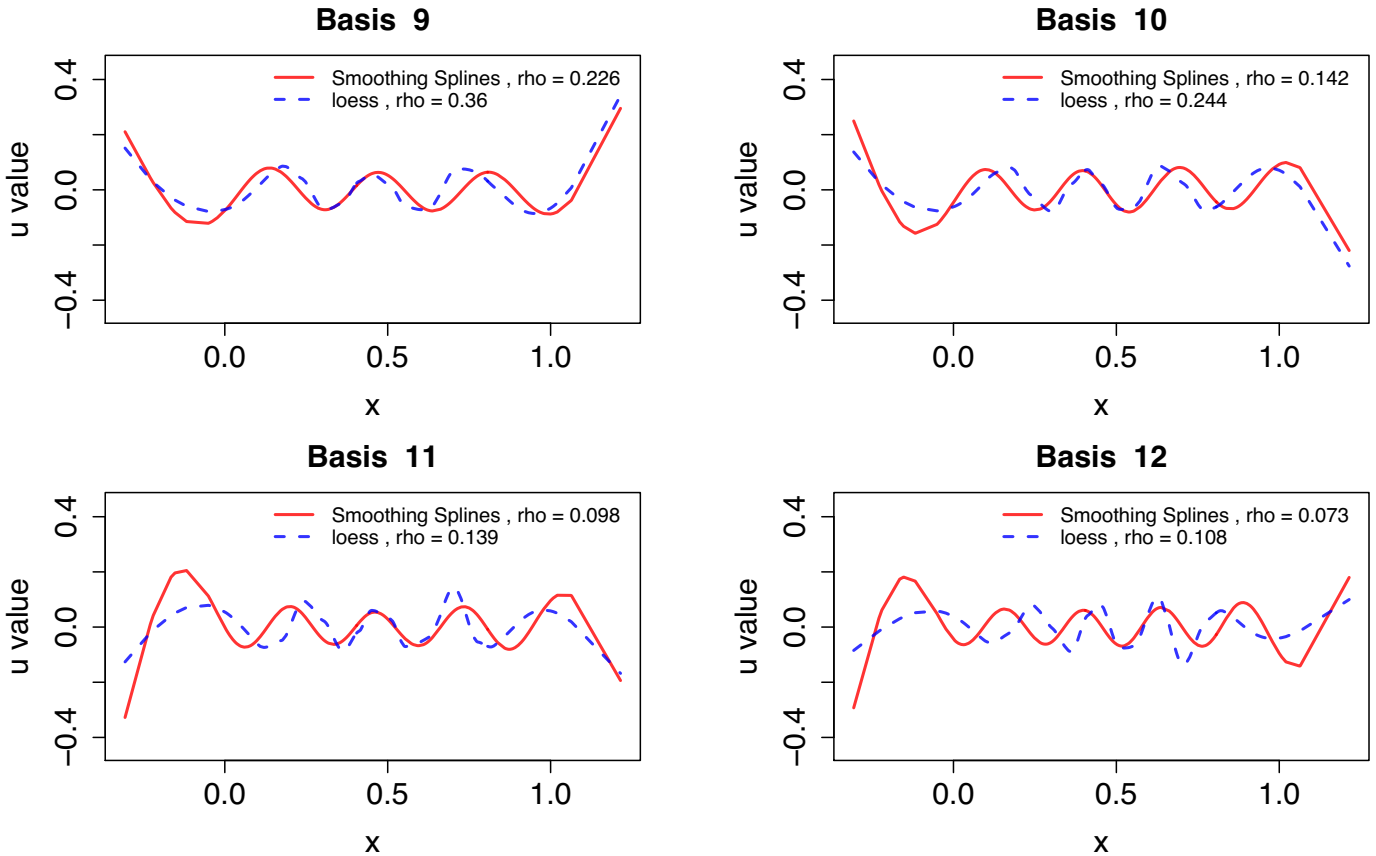
Basis Functions (first 4)



Basis Functions (5-8)



Basis Functions (9-12)



Conclusions

Even though smoothing splines were derived from a global minimization, their linear smoothing structures turns out to work in a similar way to the local model loess. We've observed the following trends in both loess and smoothing splines for our simulated data:

- **Coefficients of y:** The coefficient of y_i is higher the closer x value is to the value of x_i.
- **Singular values:** Their plot has an "elbow-shape," where singular values die off quickly. How quickly the singular values converge to zero depends on *signal to noise ratio* in the data.
- **y components $< \mathbf{V}_i$, y $>$:** Similar pattern to singular values for the same reason.
- **Basis functions:** The orthogonal basis functions increase in complexity as i increases and the higher frequency basis functions are largely obliterated by the small singular values and **y** components.

Agenda

- Introduction to Smoothing Methods
  - Polynomial Regression ✓
  - Basis Expansion ✓
- Splines
  - Cubic, Natural, and B-splines ✓
- Regularization Methods
  - Shrinkage Estimators: Ridge, LASSO, Elastic Net ✓
  - Smoothing Splines ✓
- Local Weighting
  - ~~Delta Neighbourhood and KNN~~
  - Kernel Smoothing ✓
  - Local Smoothing and LOESS ✓
- Multidimensional Smoothing (if time permits)
  - Tensor Product Bases
  - Thin-plate Splines
  - Additive Spline Model

Multidimensional Splines

- The spline models have been designed to fit a curve to a single explanatory variate x.
  - What if we have more than one explanatory variate?

- There are several ways to generalize the splines curve-fitting method, a few are:

  1) Using *tensor product basis*
     - This is a generalization of regular basis expansion models (e.g. cubic splines) to higher dimensions

  2) Using a multivariate high curvature penalty: *thin plate splines*
     - This is a generalization of smoothing splines to higher dimensions

  3) Using an *additive spline model*
     - This is another generalization of smoothing splines to higher dimensions, which imposes additivity on all components of the model.

We will not discuss the details of these methods, but simply introduce them.

1) Tensor Product Basis

- Consider $\mathbf{X} = (X_1, X_2) \in \mathbb{R}^2$, and the basis functions $h_{ik}(X_i)$, $k = 1, ..., M_i$ representing functions of coordinate X_i, i.e. two sets of basis functions one to represent coordinate X_1 and the other to represent coordinate X_2.

- The $M_1 \times M_2$ dimensional *tensor product basis* is defined by

$$g_{jk}(x_1, x_2) = h_{1j}(x_1)\, h_{2k}(x_2), \quad j = 1, ..., M_1, \ , k = 1, ..., M_2$$

and can be used for representing a two-dimensional function:

$$\mu(x_1, x_2) = \beta_0 + \sum_{j=1}^{M_1} \sum_{k=1}^{M_2} \beta_{jk} g_{jk}(x_1, x_2)$$

- The coefficients can be can be fit by least squares.

1) Tensor Product Basis

- Tensor product basis can be generalized in the same fashion to the d-dimensional case ($d > 2$), but the effective model degrees of freedom grows multiplicatively with the number of explanatory variates (curse of dimensionality).

- There are methods to select only those tensor products which are deemed necessary by least squares (e.g. MARS).

↓

Multidimensional
Additive Regression Splines

1) Tensor Product Basis : a 2-dimensional example

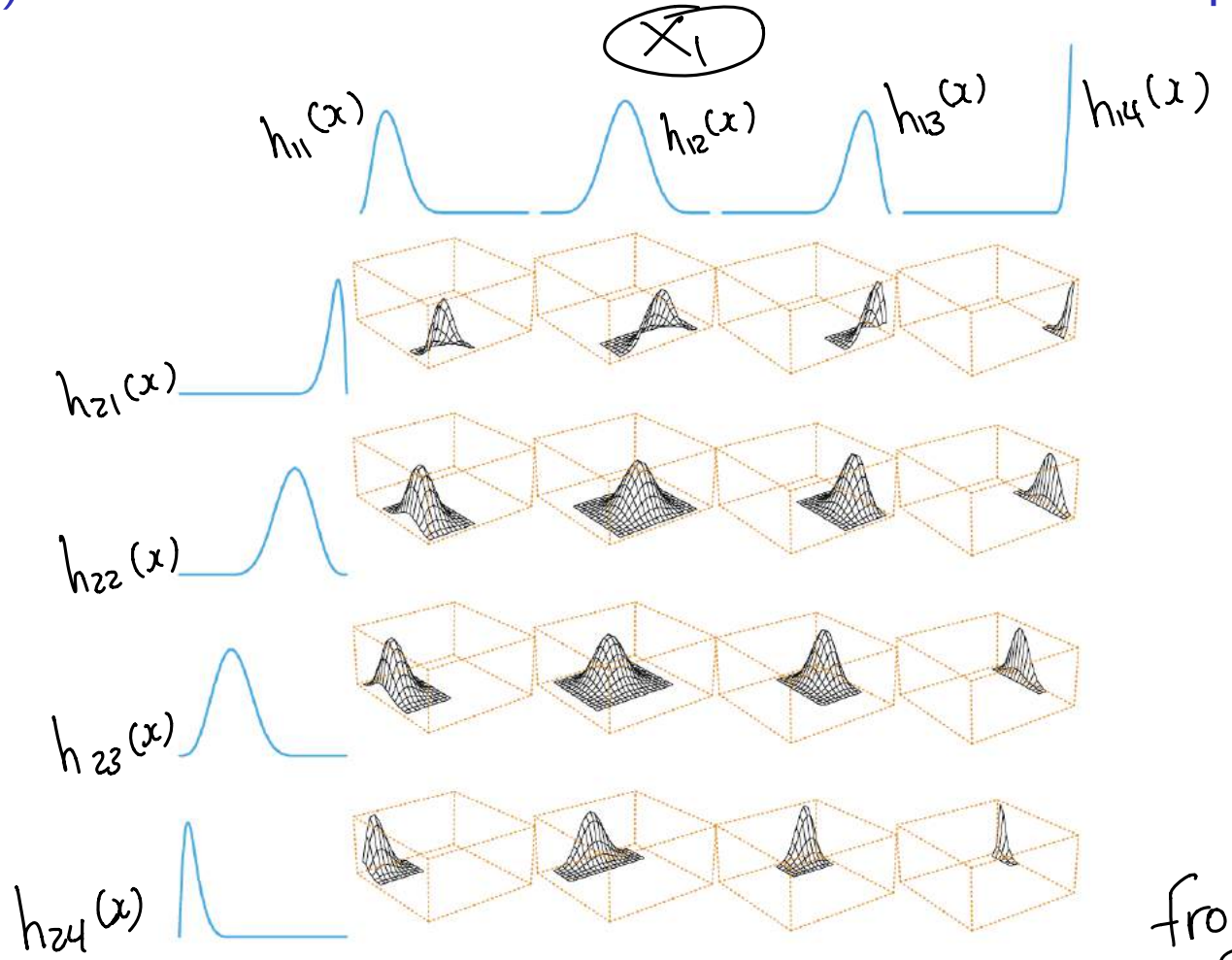


FIGURE 5.10. *A tensor product basis of B-splines, showing some selected pairs. Each two-dimensional function is the tensor product of the corresponding one dimensional marginals.*

→ from E. S. L.

2) Thin Plate Splines

- We can generalize the one-dimensional smoothing splines to higher dimensions as well.
- Consider the data $(\mathbf{x}_1\ ,\ y_1),\ldots,(\mathbf{x}_n\ ,\ y_n)$ where $\mathbf{x}_i \in \mathbb{R}^d$, and we seek a d-dimensional regression function $\mu(\mathbf{x})$.
- We can set up the problem as

$$\min_{\mu} \left\{ \sum_{i=1}^{n} \left(y_i - \mu(\mathbf{x}_i)\right)^2 + \lambda\, J[\mu] \right\}$$

  where $J[\mu]$ os am approperiate penalty function to stabilizing a function in $\mathbb{R}^d$.

2) Thin Plate Splines

- A natural generalization of the one-dimensional roughness penalty for functions on $\mathbb{R}^2$ is

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

here $d=2$

- The solution has the form

$$\mu(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^T \mathbf{x} + \sum_{i=1}^{n} \alpha_j h_j(\mathbf{x})$$

$\beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d +$

where $h_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_j\|^2 \log(\|\mathbf{x} - \mathbf{x}_j\|)$. Using this format of $\mu(\mathbf{x})$ in the penalized least squares optimization problem (previous slide), the parameters can be estimated.

3) Additive Spline Models

- The additive spline models are a restricted class of multidimensional splines.
- Suppose $\mu(\mathbf{x}) = \beta_0 + \mu_1(x_1) + ... + \mu_d(x_d)$, and that each of the functions μ_j are univariate splines.
- One natural penalty for the smoothing splines then is $J[\mu] = J[\mu_1 + \cdots + \mu_d] = \sum_{j=1}^{d} \int \mu_j''(t_j)\, dt_j$
- This method can be naturally extend to ANOVA spline decompositions.

Agenda

- Introduction to Smoothing Methods
  - Polynomial Regression ✓
  - Basis Expansion ✓
- Splines
  - Cubic, Natural, and B-splines ✓
- Regularization Methods
  - Shrinkage Estimators: Ridge, LASSO, Elastic Net ✓
  - Smoothing Splines ✓
- Local Weighting
  - ~~Delta Neighbourhood and KNN~~
  - Kernel Smoothing ✓
  - Local Smoothing and LOESS ✓
- Multidimensional Smoothing (if time permits)
  - Tensor Product Bases ✓
  - Thin-plate Splines ✓
  - Additive Spline Model ✓