LTCC Advanced Course: Introduction to Semiparametric Modelling Lecture 2

Clifford Lam

Department of Statistics
London School of Economics and Political Science

Regression with nonlinear function

- We have seen the LIDAR example where linear regression clearly cannot capture the salient features of the data.

- Generalize to the model

$$y_i = f(\mathbf{z}_i; \boldsymbol{\beta}) + \epsilon_i, \ \ i = 1, \cdots, n,$$

  where $\mathbf{z}_i$ is a vector of predictors, $\boldsymbol{\beta}$ is a vector of unknown coefficients, and $f(\mathbf{z}; \boldsymbol{\beta})$ is a known function.

- Polynomial regression and linear spline regression with **fixed number of knots** and **fixed locations of knots** are examples. They are still linear regression, since $\boldsymbol{\beta}$ is linear in the function $f(\mathbf{z}; \boldsymbol{\beta})$.

- What if the form of $f(\mathbf{z}; \boldsymbol{\beta})$ is not known? In polynomial regression, it boils down to the choice of p - the degree of polynomial to be fitted. In linear spline regression, it is reduced to finding the location of knots and the number of them.

- Or nonparametric regression, with unknown function $f(\mathbf{z})$ and model

$$y_i = f(\mathbf{z}_i) + \epsilon_i, \ \ i = 1, \cdots, n.$$

  Usually it is too general, and need some assumptions on the form of $f(\mathbf{z}_i)$.

Basis functions

- If the functional form $f(\mathbf{z};\boldsymbol{\beta})$ is not known, we assume that such a function is a linear combination of more "basic" functions - called **basis functions**.
- With one variable x, polynomial regression used

$$1, x, x^2, \ldots, x^p,$$

  with p to be determined.
- A p-th degree polynomial spline, or **truncated polynomial spline** use

$$1, x, \ldots, x^p, (x-\kappa_1)_+^p, \ldots, (x-\kappa_K)^p,$$

  with p and K needed to be determined.
- There are other common basis functions. B-spline basis and the natural spline basis are the most commonly used, along with radial basis functions. Spline basis regression will be treated under a unified framework to be introduced later.

B-spline basis

One commonly used basis is called the B-spline basis. One advantage of such basis is the numerical stability compared to e.g. polynomial spline. However, a p-th degree B-spline is actually equivalent to a p-th degree polynomial spline if they have the same knot locations. For instance, if

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^p & (x_1-\kappa_1)_+^p & \cdots & (x_1-\kappa_K)_+^p \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^p & (x_n-\kappa_1)_+^p & \cdots & (x_n-\kappa_K)_+^p \end{pmatrix},$$

representing the design matrix of a p-th degree polynomial spline fit with predictor variable x, then there exists an invertible matrix $\mathbf{L}$ such that the design matrix $\mathbf{X}_B$ for the B-spline fit can be written as

$$\mathbf{X}_B = \mathbf{X}\mathbf{L}.$$

B-spline basis : example

```
library(splines)
plot(bs(1:400, df=5)[,1], type="l", ylab="", xlab="", col=1)
lines(bs(1:400, df=5)[,2], type="l", ylab="", xlab="", col=2)
lines(bs(1:400, df=5)[,3], type="l", ylab="", xlab="", col=3)
lines(bs(1:400, df=5)[,4], type="l", ylab="", xlab="", col=4)
lines(bs(1:400, df=5)[,5], type="l", ylab="", xlab="", col=5)
```

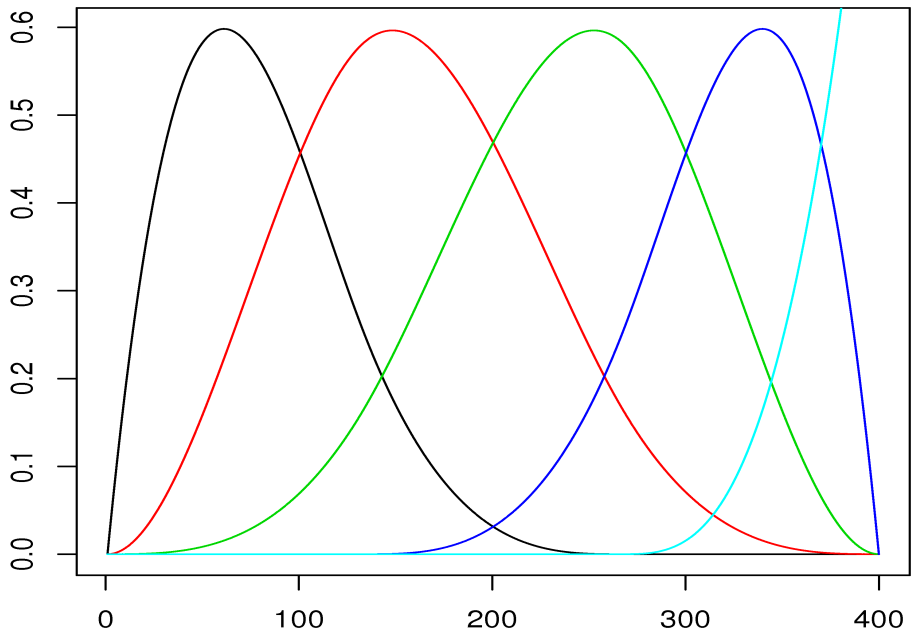


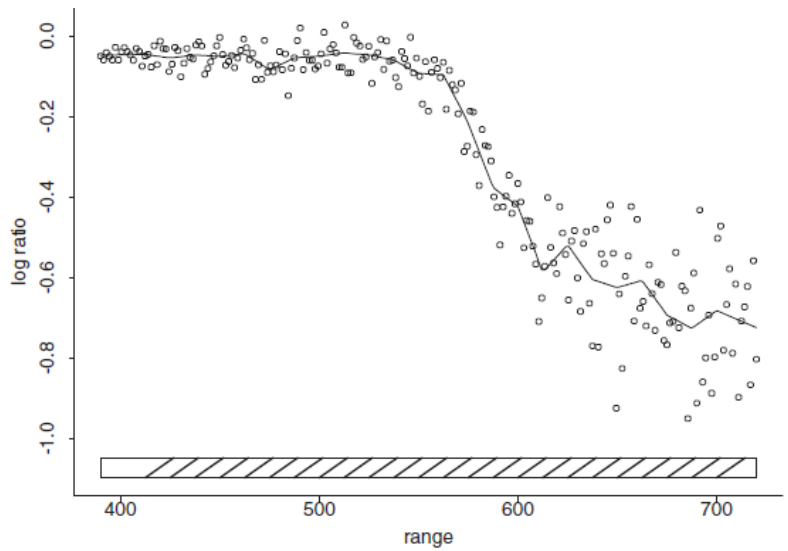
Figure: Fifth degree B-spline basis functions

Linear spline basis

- In the LIDAR data example from previous lecture, we used the following linear spline basis functions:

$$1, x, (x-\kappa_1)_+, (x-\kappa_2)_+, \ldots, (x-\kappa_K).$$

- Essentially, we use **piecewise linear** functions to build the function $f(\mathbf{z};\beta)$, with the knots $\kappa_1, \cdots, \kappa_K$ representing potential discontinuity in the first derivative of the curve.
- With K too large, we are **overfitting** the data. E.g. interpolation, setting all data x_i as knots for $i = 1, \cdots, n$.
- With K too small, salient features may not be captured. E.g. linear regression on the LIDAR data.



- LIDAR data fitted with linear spline regression. Knots at $400, 412.5, \cdots, 700$.
- Too many knots essentially overfitted the model.

Penalized spline regression

- Too many knots makes the resulting curve looks more wiggly. On the other hand, since

$$E(y|x) = \beta_0 + \beta_1 x + \cdots + \beta_p x^p + \sum_{k=1}^{K} b_k (x - \kappa_k)_+^p,$$

if $b_k = 0$ for all k then the fit becomes a p-th degree polynomial fit, which will not be wiggly provided p is of a suitable order.

- The magnitude of the b_k's control the **roughness** of the curve. If we want a smoother fit, then we want the b_k's to be small.
- One way to achieve this is to set the constraint $\sum b_k^2 \leq C$ where C is a constant. Hence we need to solve

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \text{ subject to } \sum_{k=1}^{K} b_k^2 \leq C,$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \cdots, \beta_p, b_1, \cdots, b_K)^T$, and $\mathbf{X}$ is the design matrix defined on page 4.

Penalized spline regression

- By Langrange multiplier argument, the problem can be written as

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \boldsymbol{\beta}^T \mathbf{D} \boldsymbol{\beta},$$

  where λ is called a **smoothing parameter**, and

$$\mathbf{D} = \begin{pmatrix} \mathbf{0}_{p+1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_K \end{pmatrix}$$

  is a diagonal matrix with 0 on the first $p+1$ diagonal elements, and 1 on the rest.

- This is a ridge regression type problem, with solution

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X} + \lambda \mathbf{D})^{-1} \mathbf{X}^T \mathbf{y}.$$

- When $\lambda = 0$, it is a p-th degree polynomial spline using all K knots. When $\lambda \to \infty$, it is a p-th degree polynomial regression fit.

Penalized spline regression : LIDAR example

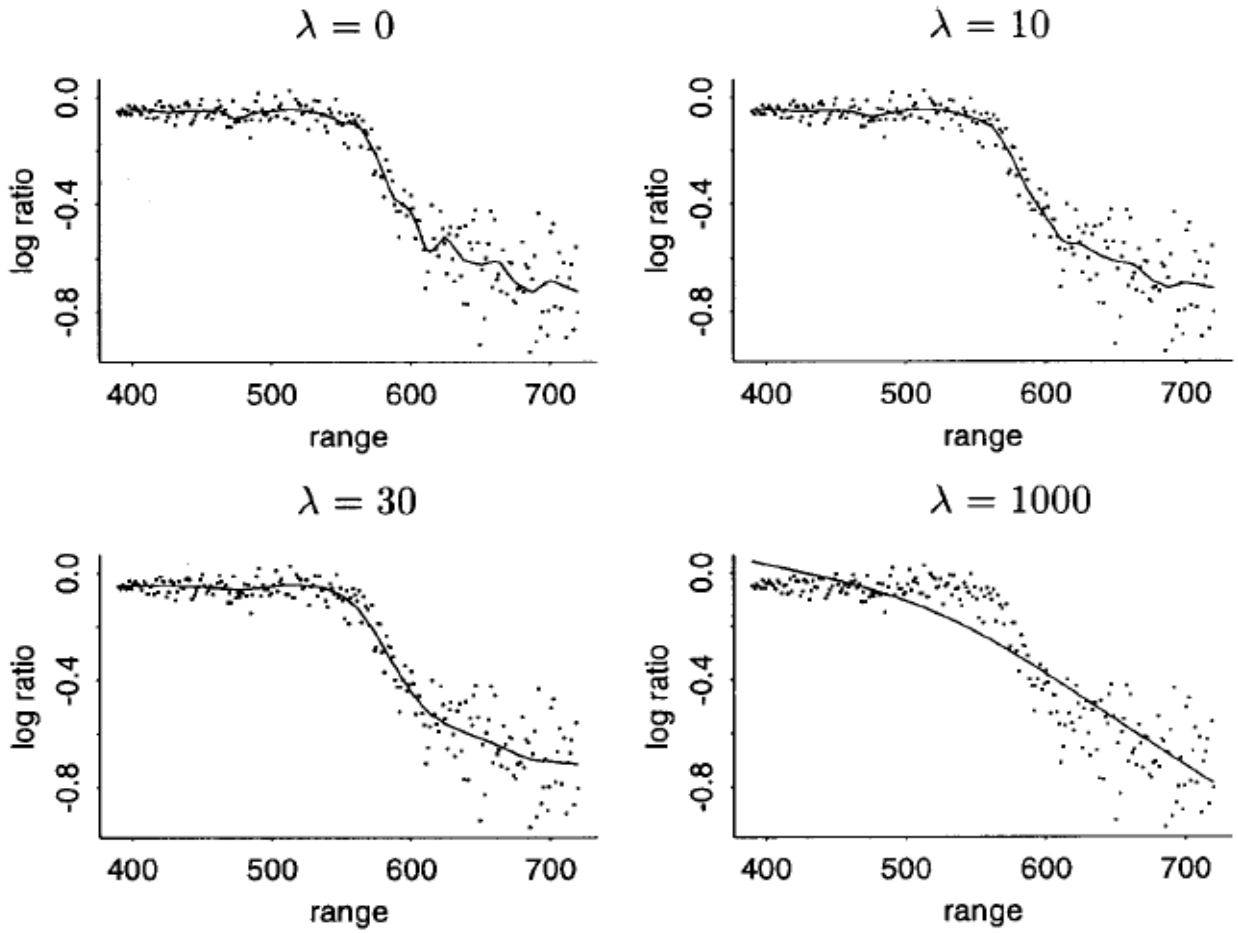


Figure: Linear penalized spline regression for the LIDAR data. 24 knots are used.

Penalized spline regression - the general form

- Suppose $\mathbf{B}(x) = (B_1(x), \cdots, B_N(x))^T$ is a vector of spline basis functions. E.g. in a p-th degree polynomial spline, $B_1(x) \equiv 1$, $B_2(x) = x$, and $B_N(x) = (x - \kappa_K)_+^p$, with $N = p + 1 + K$. The design matrix can then be written as

$$\mathbf{X} = \begin{pmatrix} \mathbf{B}(x_1)^T \\ \vdots \\ \mathbf{B}(x_n)^T \end{pmatrix},$$

  and $E(y|x) = f(x) = \mathbf{B}(x)^T \boldsymbol{\beta}$.

- A p-th degree polynomial spline is shrunk towards a p-th degree polynomial as $\lambda \to \infty$. A more flexible approach is to use a penalty of the form

$$\lambda \int_a^b [f^{(q+1)}(x)]^2 \, dx.$$

  As $\lambda \to \infty$, we need $f^{(q+1)}(x) = 0$ essentially, so that the resulting fit is a q-th degree polynomial. Since $f(x) = \mathbf{B}(x)^T \boldsymbol{\beta}$, the penalty can be written as $\boldsymbol{\beta}^T \mathbf{D} \boldsymbol{\beta}$, where

$$\mathbf{D} = \int_a^b \mathbf{B}^{(q+1)}(x) [\mathbf{B}^{(q+1)}(x)]^T \, dx.$$

- Solution has the same form as before, with the only change being the definition of $\mathbf{D}$.

Penalized spline regression - the big picture

There are several choices to be made when applying penalized splines:

1. The spline model - the degree p, knot locations κ_k and total number of knots K.
2. The penalty function.
3. The basis functions. Interpretability and computability are both issues. B-spline basis has more numerical stability, but interpretability is very much hindered, while polynomial spline basis has more interpretability but not computability. But, they are equivalent to one another.

With 2 and 3, the penalty matrix $\mathbf{D}$ is fixed.

- For the choice of the number of knots, there are actually automatic algorithm (myopic algorithm) for this purpose. However, in practice the result is usually not too sensitive to the choice of K, provided that K is reasonably large (20 to 40 for large data set).
- For knots location, it can be prefixed, or choose them by visual inspection.
- Choice of λ is much more important, and we introduce the method of CV, GCV, AIC or Mallow's C_p soon.

Linear Smoothers

- In penalized spline regression, the solution for β is $\hat{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{D})^{-1}\mathbf{X}^T\mathbf{y}$, so

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{D})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{S}_\lambda\mathbf{y},$$

  where $\mathbf{S}_\lambda = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{D})^{-1}\mathbf{X}^T$. This is analogous to the hat matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$, and is called the **smoother matrix**.

- For the class of models with $\hat{\mathbf{y}} = \mathbf{L}\mathbf{y}$ where $\mathbf{L}$ is a matrix independent of the data $\mathbf{y}$, the matrix $\mathbf{L}$ is called **a linear smoother**. Linear regression and penalized spline regression all have an associated smoother matrix. The latter is so assuming λ is a fixed constant.

Error of an estimator

- The error of an estimator $\hat{f}(x)$ on $f(x)$ can be assessed by the **mean square error (MSE)**, defined by $\text{MSE}\{\hat{f}(x)\} = E[\{\hat{f}(x) - f(x)\}^2]$. The following can be derived, where $\text{bias}\{\hat{f}(x)\} = E(\hat{f}(x)) - f(x)$:

$$\text{MSE}\{\hat{f}(x)\} = \text{var}\{\hat{f}(x)\} + \text{bias}^2\{\hat{f}(x)\}.$$

- Yet MSE is only measuring on a point x. It is common to measure error globally across x. One possibility is the **mean integrated squared error (MISE)**:

$$\text{MISE}\{\hat{f}(\cdot)\} = \int_{\chi} \text{MSE}\{\hat{f}(x)\}\, dx.$$

The mean summed squared error (MSSE)

- Instead of MISE which depends on χ, a simpler alternative is to use MSSE:

$$\text{MSSE}\{\hat{f}(\cdot)\} = E\sum_{i=1}^{n}\{\hat{f}(x_i) - f(x_i)\}^2.$$

- It has simple expression after simplification. Denote $\hat{\mathbf{f}} = (\hat{f}(x_1), \ldots, \hat{f}(x_n))^T$. Similarly for $\mathbf{f}$. For linear smoothers with smoother matrix $\mathbf{L}$, we have $\hat{\mathbf{f}} = \mathbf{L}\mathbf{y}$. Then by **noting that** $E(\mathbf{y}) = \mathbf{f}$ **and assuming var**$(\mathbf{y}) = \sigma^2\mathbf{I}$,

$$\begin{aligned}
\text{MSSE}(\hat{f}(\cdot)) &= \text{MSSE}(\hat{\mathbf{f}}) = E\|\hat{\mathbf{f}} - \mathbf{f}\|_2^2 \\
&= E\|\hat{\mathbf{f}} - E(\hat{\mathbf{f}}) + E(\hat{\mathbf{f}}) - \mathbf{f}\|_2^2 \\
&= E\|\hat{\mathbf{f}} - E(\hat{\mathbf{f}})\|_2^2 + 2E[\{\hat{\mathbf{f}} - E(\hat{\mathbf{f}})\}^T\{E(\hat{\mathbf{f}}) - \mathbf{f}\}] + E\|E(\hat{\mathbf{f}}) - \mathbf{f}\|_2^2 \\
&= E\{\hat{\mathbf{f}} - E(\hat{\mathbf{f}})\}^T\{\hat{\mathbf{f}} - E(\hat{\mathbf{f}})\} + \|\mathbf{L}\mathbf{f} - \mathbf{f}\|_2^2 \\
&= \text{tr}[E\{(\mathbf{L}\mathbf{y} - \mathbf{L}\mathbf{f})(\mathbf{L}\mathbf{y} - \mathbf{L}\mathbf{f})^T\}] + \|(\mathbf{L} - \mathbf{I})\mathbf{f}\|_2^2 \\
&= \text{tr}[\mathbf{L}\text{var}(\mathbf{y})\mathbf{L}^T] + \|(\mathbf{L} - \mathbf{I})\mathbf{f}\|_2^2 \\
&= \sigma^2\text{tr}(\mathbf{L}\mathbf{L}^T) + \|(\mathbf{L} - \mathbf{I})\mathbf{f}\|_2.
\end{aligned}$$

Rank of a linear smoother $\mathbf{L}$

- Classical smoothing spline uses n basis functions, whereas linear penalized splines with K knots uses $K+2$ basis functions for estimating $E(y|x)$.
- Let $\mathbf{L}$ be a smoother matrix with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$, and corresponding eigenvectors $\mathbf{v}_1, \cdots, \mathbf{v}_n$, which form a basis for $\mathbb{R}^n$. Then there exists constants $\alpha_1, \cdots, \alpha_n$ such that

$$\mathbf{y} = \sum_{j=1}^{n} \alpha_j \mathbf{v}_j.$$

Hence pre-multiplying $\mathbf{L}$ on both sides,

$$\hat{\mathbf{y}} = \sum_{j=1}^{n} \alpha_j \mathbf{L}\mathbf{v}_j = \sum_{j=1}^{n} \alpha_j \lambda_j \mathbf{v}_j.$$

For a **low-rank smoother** $\mathbf{L}$ which uses considerably less than n basis functions, many eigenvalues of $\mathbf{L}$ are close to or exactly 0. The above expression means that the main part of the smoothing spline fit comes from the eigenvectors which have non-zero eigenvalues, and other parts contribute just little (small λ_j) or have no contribution at all ($\lambda_j = 0$).

- For n in thousands or higher, reduction in computational cost is huge using low-rank smoother.

Degree of freedom of a smoother $\mathbf{L}$

- We have seen that $\mathbf{S}_\lambda$ is a generalization of the hat matrix $\mathbf{H}$ in linear regression, in the sense that

$$\hat{\mathbf{y}} = \mathbf{S}_\lambda \mathbf{y}.$$

- Since $\text{tr}(\mathbf{H}) = $ total number of parameters fitted, and is called the degrees of freedom, we define

$$t(\lambda) = \text{tr}(\mathbf{S}_\lambda)$$

to be the **degrees of freedom of the fit corresponding to the smoothing parameter** λ.

- It has the rough interpretation as the **equivalent number of parameters**, and roughly a scatterplot smooth with ν degrees of freedom summarizes the data to about the same extent as a $(\nu - 1)$-degree polynomial.
- We use the notation

$$df_{\text{fit}} = \text{tr}(\mathbf{S}_\lambda).$$

E.g. for a penalized spline,

$$df_{\text{fit}} = \text{tr}\{(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{D})^{-1}\mathbf{X}^T\mathbf{X}\}.$$

- Can show easily that $p+1 < df_{\text{fit}} < p+1+K$ for a p-th degree polynomial spline.

Residual degree of freedom

- We define the residual degree of freedom of the fit to be

$$df_{\text{res}} = n - 2\text{tr}(\mathbf{S}_\lambda) + \text{tr}(\mathbf{S}_\lambda \mathbf{S}_\lambda^T).$$

  E.g. for linear regression where $\mathbf{S}_\lambda$ is replaced by $\mathbf{H}$, we have

$$df_{\text{res}} = n - 2p + p = n - p,$$

  where p is the degree of freedom for linear regression.

- In exercise 2, you will show that

$$E(\text{RSS}) = E\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 = \|(\mathbf{S}_\lambda - \mathbf{I})\mathbf{f}\|_2^2 + \sigma^2 df_{\text{res}}.$$

  Hence if the bias term is negligible, $\text{RSS}/df_{\text{res}}$ is indeed an unbiased estimator of σ^2, analogous to linear regression.

Comparing df_{fit} and $n - df_{\text{res}}$

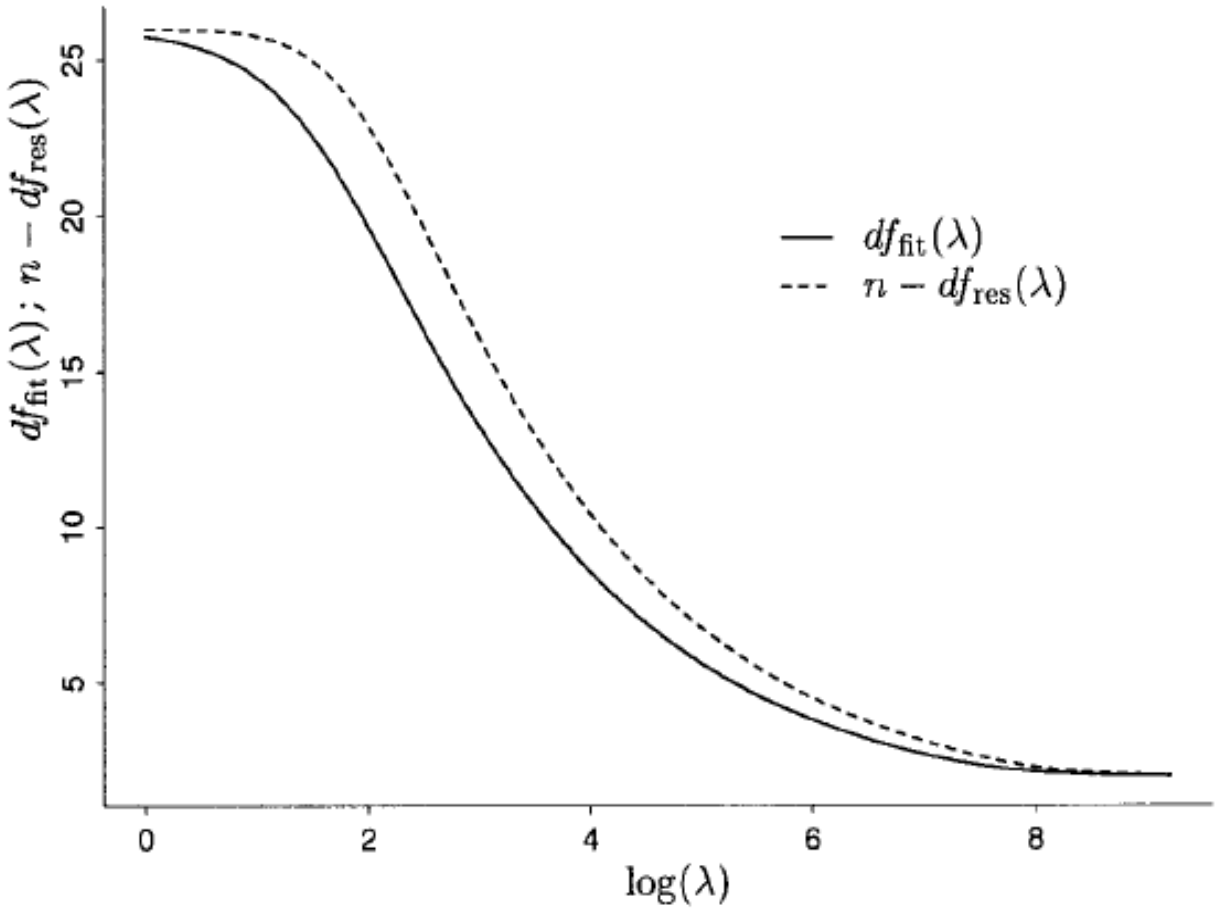


Figure: Comparison of df_{fit} and $n - df_{\text{res}}$ for 24-knot linear spline fits to the LIDAR data.

Cross Validation (CV) for choosing λ

- The residual sum of squares RSS is a measure of goodness of fit. However, it is always the case that RSS is minimized when $\lambda = 0$. That is, when there is no penalization. The resulting fit is unacceptable as explained earlier.
- The cross validation criterion (CV) is defined as

$$\mathsf{CV}(\lambda) = \sum_{i=1}^{n} \{y_i - \hat{f}_{-i}(x_i; \lambda)\}^2,$$

where $\hat{f}_{-i}$ denotes the estimator applied to the data but with (x_i, y_i) deleted. This is a "leave-one-out" criterion which gets around the problem inherited in RSS. It attains a minimum value for some $\lambda > 0$.

- For large n, computational cost can be high if we directly compute $\mathsf{CV}(\lambda)$. However, for many kinds of smoothers, it can be shown that approximately,

$$\mathsf{CV}(\lambda) = \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - (\mathbf{S}_\lambda)_{ii}} \right)^2.$$

Hence ordinary residuals can be used to compute CV, reducing hugely the computational cost.

CV vs RSS

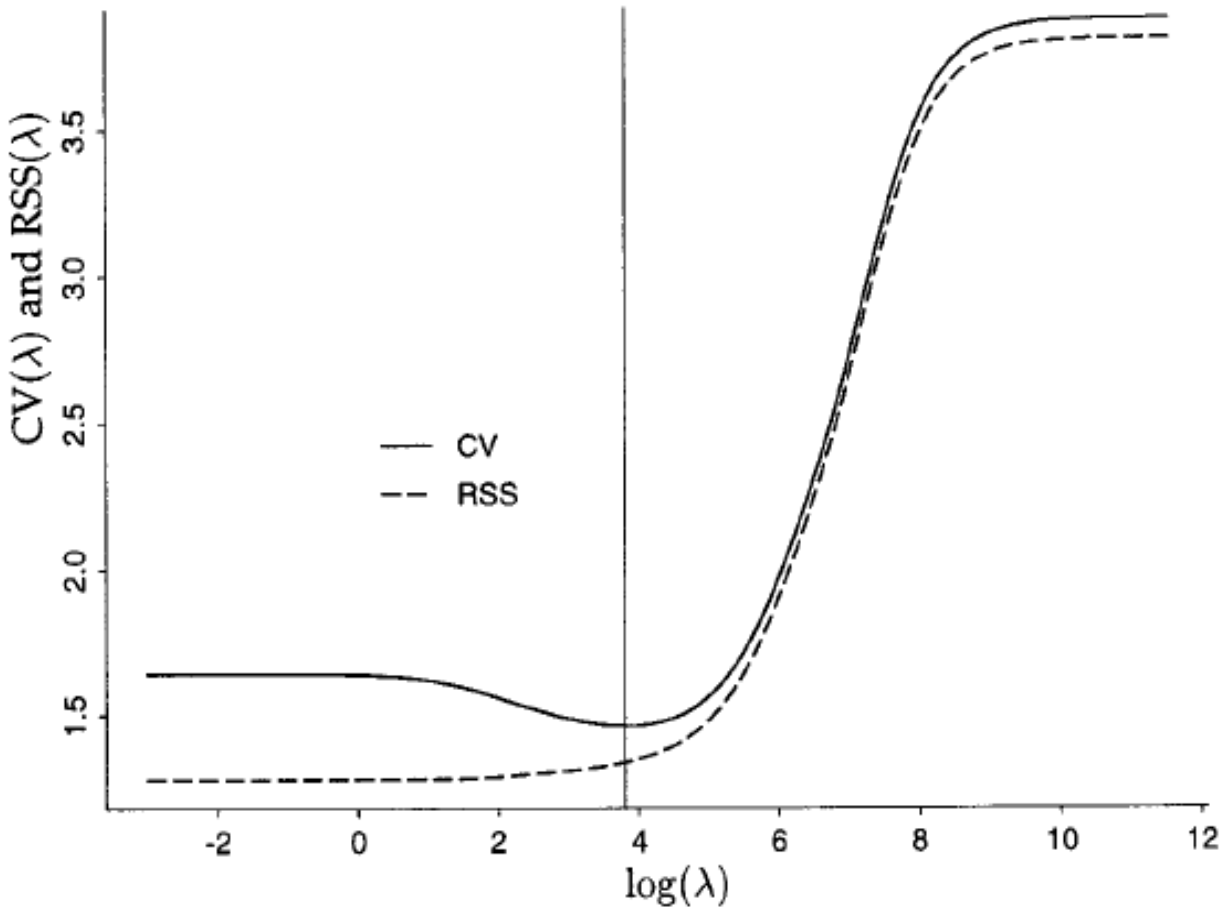


Figure: CV and RSS curves for the LIDAR data using 24-knot linear regression splines.

Generalized Cross Validation (GCV) for choosing λ

- The GCV replaces $(\mathbf{S}_\lambda)_{ii}$ in the CV formula by the average

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbf{S}_\lambda)_{ii} = \frac{1}{n}\mathsf{tr}(\mathbf{S}_\lambda).$$

  It is thus much quicker to calculate.

- The GCV criterion is

$$\mathsf{GCV}(\lambda) = \sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - n^{-1}\mathsf{tr}(\mathbf{S}_\lambda)}\right)^2 = \frac{\mathsf{RSS}(\lambda)}{(1 - n^{-1}\mathsf{tr}(\mathbf{S}_\lambda))^2}.$$

- For the LIDAR data example, the CV and GCV curves are extremely close together, and attains the same minimum at the same λ.
- Other methods like Mallow's C_p and AIC exist also. We will skip them in the lecture, but they may appear in exercises.

Local polynomial regression

- At each point $x \in \chi$, we fit a weighted least square regression line to the data.

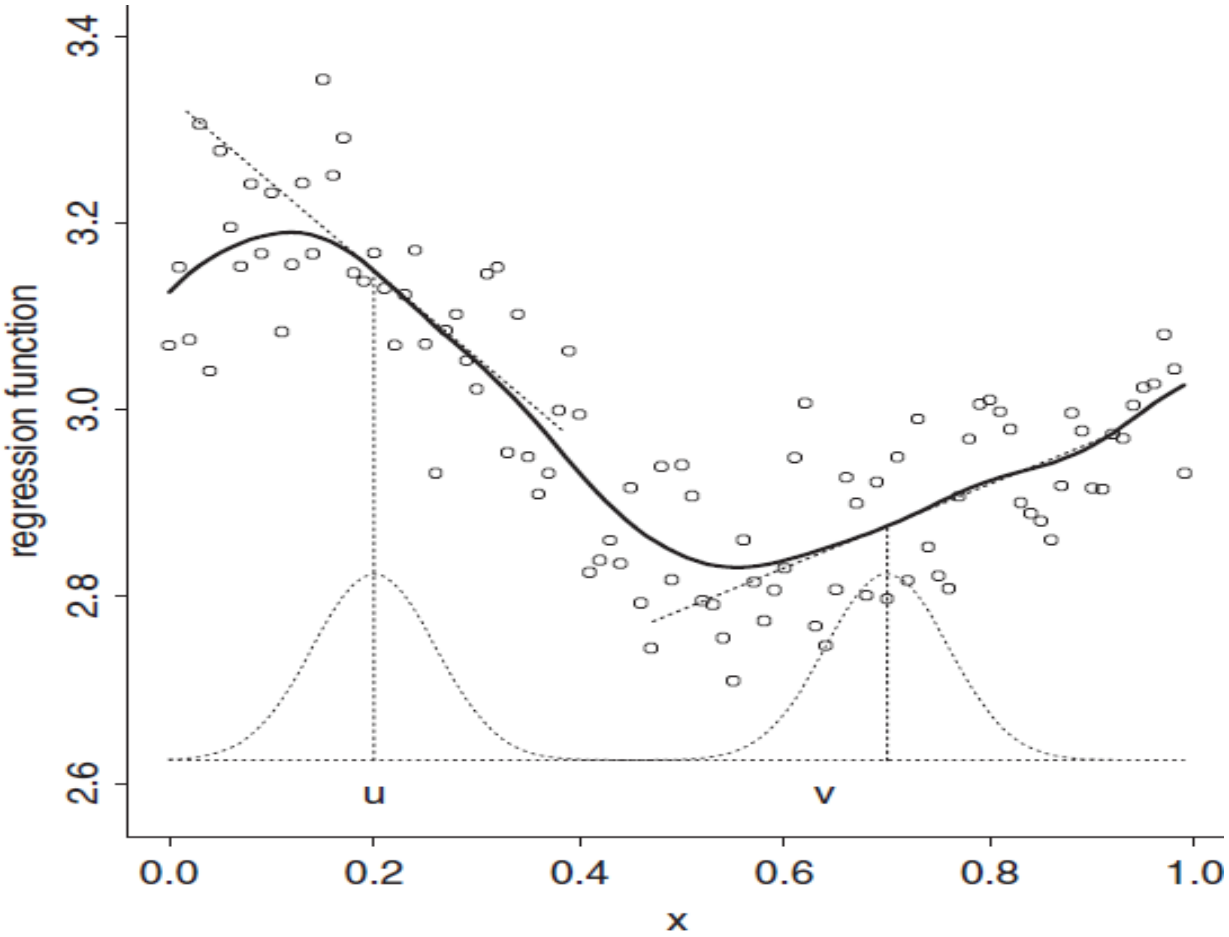


Figure: Cubic fits of the data. Dotted curve are kernel weights for points u and v.

Local polynomial regression

- When using least squares on the model $y = f(x) + \epsilon$, we minimize

$$\sum_{i=1}^{n} (y_i - f(x_i))^2.$$

- At a point $x \in \chi$, using Taylor's expansion on $f(x_i)$ at the point x up to $p+1$ terms, **if x is close to x_i**, we have

$$f(x_i) \approx f(x) + f'(x)(x_i - x) + \cdots + f^{(p)}(x)(x_i - x)^p.$$

- The $f^{(j)}(x)$'s should be approximately constants **for a small range of values of x close to x_i**, say $\beta_j \approx f^{(j)}(x)$.
- Instead of using least squares, we use weighted least squares with weights provided by the **kernel function** $K(\cdot)$, so that for those x_i far away from x, their contribution is weighted to be very small:

$$\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1(x_i - x) - \cdots - \beta_p(x_i - x)^p) K\left(\frac{x_i - x}{h}\right).$$

- In essence we are fitting a polynomial regression at the point x with those x_i's which are close to x; all other x_i's far from x are contributing small to nothing.

Local polynomial regression

- The problem becomes a standard weighted least square problem, with solution

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}_x^T \mathbf{W}_x \mathbf{X})^{-1} \mathbf{X}_x^T \mathbf{W}_x \mathbf{y},$$

where $\mathbf{W}_x = \text{diag}\left(K\left(\frac{x_1 - x}{h}\right), \ldots, K\left(\frac{x_n - x}{h}\right)\right)$, and

$$\mathbf{X}_x = \begin{pmatrix} 1 & x_1 - x & \cdots & (x_1 - x)^p \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_n - x & \cdots & (x_n - x)^p \end{pmatrix}.$$

- The estimators $\hat{\beta}_0, \cdots, \hat{\beta}_p$ are estimating $f(x), \cdots, f^{(p)}(x)$ respectively.
- We need to choose $K(\cdot)$, the kernel function. Usually results will not be too sensitive to this choice for a wide variety of standard kernel functions.
- The parameter h is called the **bandwidth**. The smaller it gets, the more local we are looking at around x. It can be chosen by CV or GCV methods introduced earlier.

Local polynomial regression

- The Nadaraya-Watson estimator takes $p = 0$, i.e. a local constant fit to the data. In practice usually $p \geq 1$ performs much better. Theoretically, we should take p to be odd in order that the bias in the boundary be reduced. In practice, $p = 2$ also works well in many circumstances when we are concerned with prediction inside the boundaries of χ.
- However, if we want extrapolation for prediction outside the boundaries, we should go back to odd p, like local linear or local cubic fits, since bias in the boundaries play an important role now.
- Advantages of local polynomial regression with $p \geq 1$ includes easy theoretical analysis, automatically correct bias in the boundary as compared to local constant fit (automatic kernel carpentry, you will see problems related to this point in exercise 2). Also compared to splines methods, there are less parameters to choose. Essentially only the bandwidth h needed to be chosen when we are doing e.g. local linear regression with a certain type of kernel.
- Disadvantages include difficult interpretation of the regression function, and no explicit functional form for the final estimator of $f(\cdot)$. It also takes more time to compute than splines. And it needs denser data than splines for accurate estimation, as it looks at the data "locally".

Local polynomial regression - example

```
x=-100:150/50
y = 0.4*x^3 - 1.5*x^2 + 0.5 + rnorm(251)
plot(x,y)
lines(x, 0.4*x^3 - 1.5*x^2 + 0.5)

y.lo1 = loess(y ~ x, degree=1, span = 0.75)
lines(x, y.lo1$fit, col=2)
y.lo2 = loess(y ~ x, degree=2, span = 0.75)
lines(x, y.lo2$fit, col=3)
```

Local polynomial regression - example

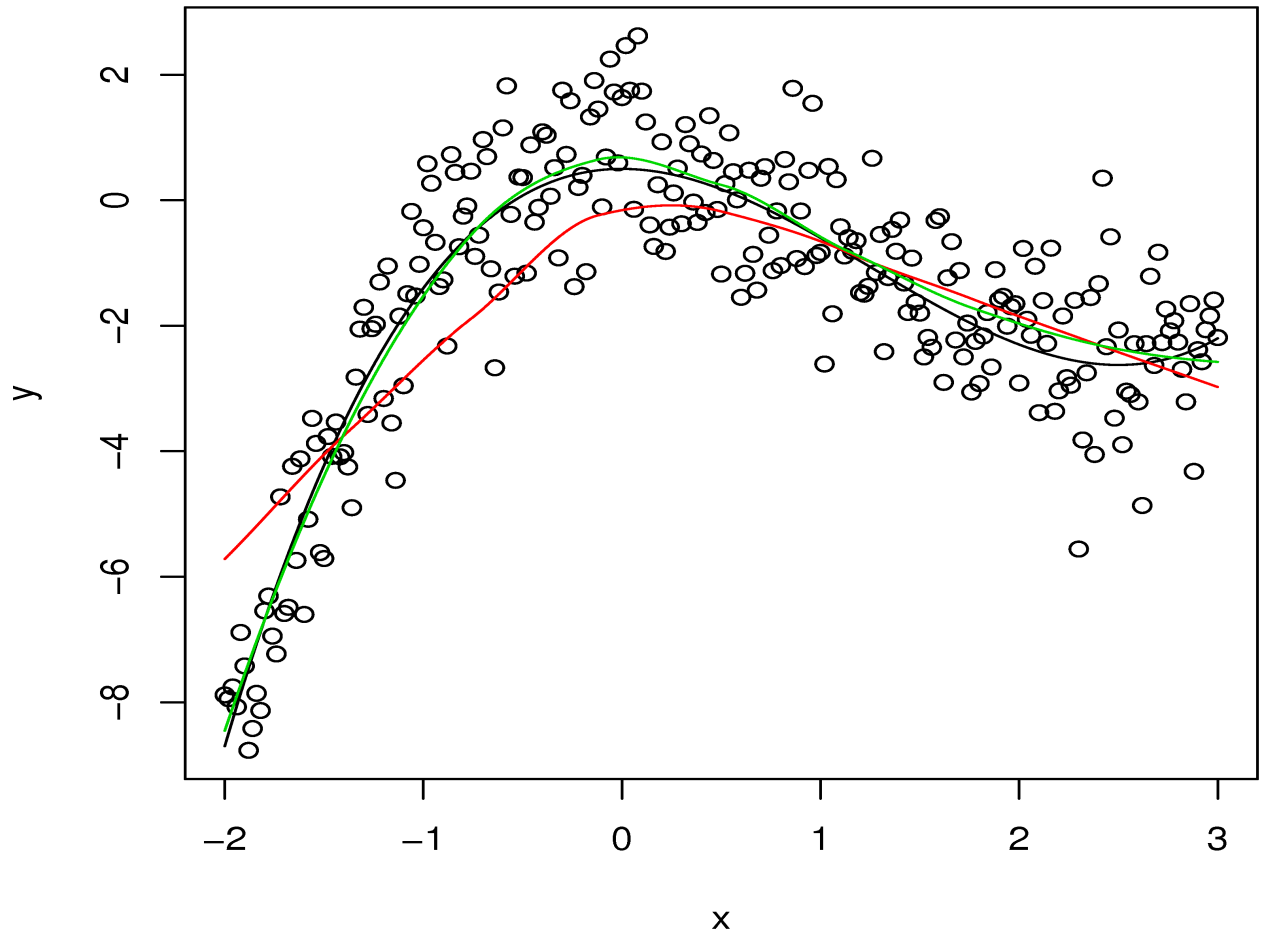


Figure: Local linear (red) and local quadratic (green) fits. Black line is the true underlying function.

Local polynomial regression - example

```
x=-100:150/50
y = 0.4*x^3 - 1.5*x^2 + 0.5 + rnorm(251)
plot(x,y)
lines(x, 0.4*x^3 - 1.5*x^2 + 0.5)

y.lo1 = loess(y ~ x, degree=1, span = 0.2)
lines(x, y.lo1$fit, col=2)
y.lo2 = loess(y ~ x, degree=2, span = 0.2)
lines(x, y.lo2$fit, col=3)
```

Local polynomial regression - example

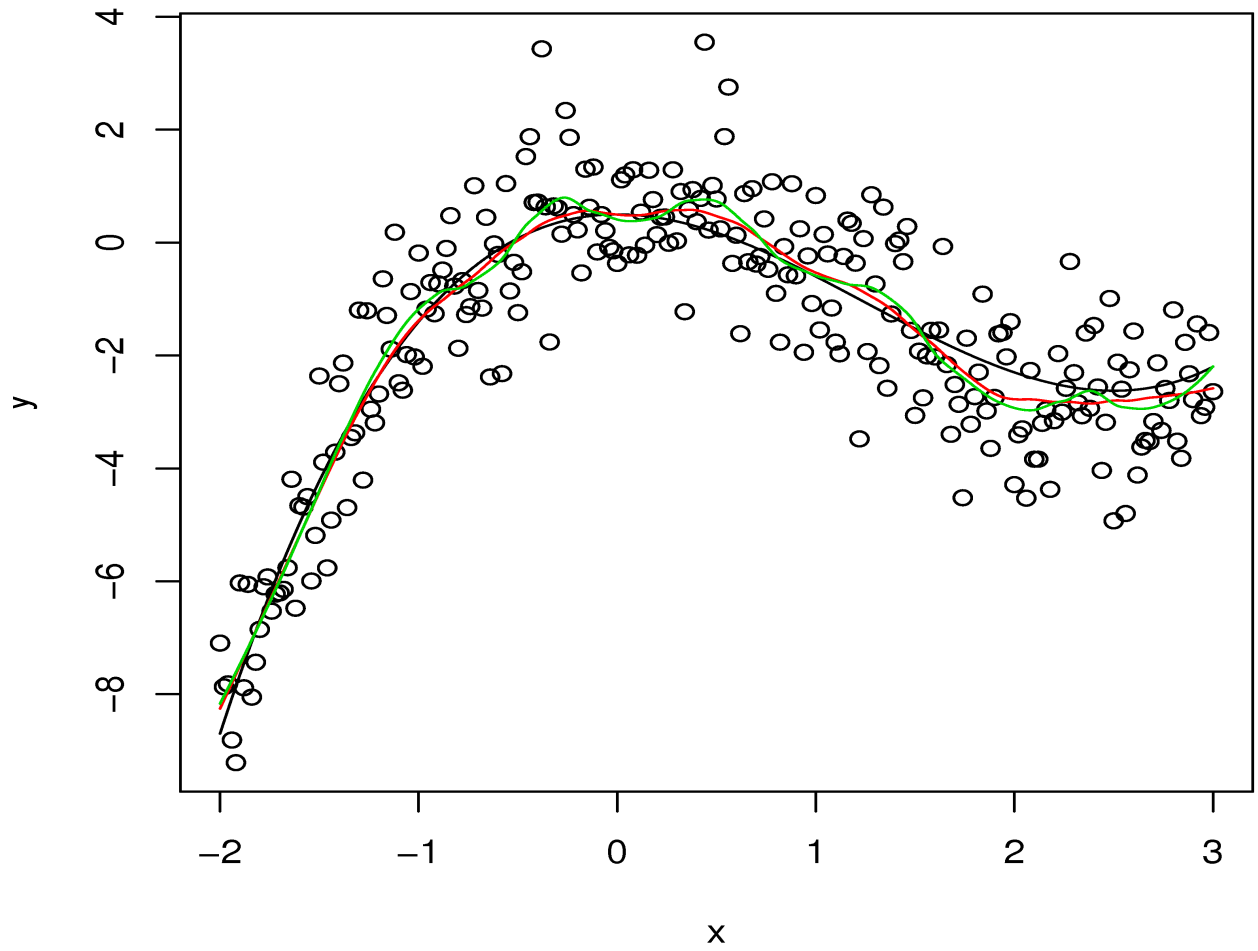


Figure: Local linear (red) and local quadratic (green) fits. Black line is the true underlying function.

Local polynomial regression - bias and variance

- This is from Fan and Gijbels (1996), Local Polynomial Modelling and Its Applications.
- Define

$$\mu_j = \int u^j K(u)\, du \quad \nu_j = \int u^j K^2(u)\, du$$

Also define

$$S = (\mu_{j+\ell})_{0\le j,\ell\le p}, \quad c_p = (\mu_{p+1}, \ldots, \mu_{2p+1})^T$$
$$S^* = (\nu_{j+\ell})_{0\le j,\ell\le p}, \quad \tilde{c}_p = (\mu_{p+2}, \ldots, \mu_{2p+2})^T.$$

Finally, let $\mathbf{e}_k$ be the unit column vector with 1 at the k-th position.

Local polynomial regression - bias and variance

Theorem

Let f_x be the density function of x. Assume that $f_x(x_0) > 0$ and $f_x(\cdot)$, $f^{(p+1)}(\cdot)$ and $\sigma^2(\cdot)$ are continuous in a neighborhood of x_0. Further, assume that $h \to 0$ and $nh \to \infty$. Then the asymptotic conditional variance of $\widehat{f^{(k)}}(x_0)$ is given by

$$\text{var}(\widehat{f^{(k)}}(x_0)|\mathbb{X}) = \mathbf{e}_{k+1}^T S^{-1} S^* S^{-1} \mathbf{e}_{k+1} \frac{k!^2 \sigma^2(x_0)}{f_x(x_0) n h^{1+2k}} + o_P\left(\frac{1}{nh^{1+2k}}\right).$$

The asymptotic conditional bias for $p-k$ odd is given by

$$\text{Bias}(\widehat{f^{(k)}}(x_0)|\mathbb{X}) = \mathbf{e}_{k+1}^T S^{-1} c_p \frac{k!}{(p+1)!} f^{(p+1)}(x_0) h^{p-k+1} + o_P(h^{p-k+1}).$$

For $p-k$ even the asymptotic conditional bias is

$$\text{Bias}(\widehat{f^{(k)}}(x_0)|\mathbb{X}) = \mathbf{e}_{k+1}^T S^{-1} \tilde{c}_p \frac{k!}{(p+2)!} \left\{ f^{(p+2)}(x_0) + (p+2) f^{(p+1)}(x_0) \frac{f_x'(x_0)}{f_x(x_0)} \right\} h^{p-k+2} + o_P(h^{p-k+2}),$$

provided that $f_x'(\cdot)$ and $f^{(p+2)}(\cdot)$ are continuous in a neighborhood of x_0 and $nh^3 \to \infty$.

Local polynomial regression - bias and variance

- Theoretical differences for the bias term between $p-k$ is odd and $p-k$ is even.
- It turns out that theoretically, polynomial fits with $p-k$ is odd outperform those with $p-k$ even.
- Suppose $k=0$ - we are focusing on the regression function $f(\cdot)$. Then define $V_p = \mathbf{e}_1^T S^{-1} S^* S^{-1} \mathbf{e}_1$, we can show

$$V_0 = V_1 = \nu_0, \quad V_2 = V_3 = \frac{\mu_4^2 \nu_0 - 2\mu_2\mu_4\nu_2 + \mu_2^2\nu_4}{(\mu_4 - \mu_2^2)^2}.$$

- Variance increases only from odd order to consecutive even order, but not from even order to consecutive odd order.
- Higher order fit reduces bias. Hence odd order fit is better than its consecutive lower even order fit, because bias is reduced without variance changing asymptotically. In particular, local linear fit is better than local constant fit for the regression function.
- For $k>0$, similar arguments can apply.

Local polynomial regression - Equivalent kernel

- Since $\hat{\boldsymbol{\beta}} = (\mathbf{X}_x \mathbf{W}_x \mathbf{X}_x)^{-1} \mathbf{X}_x^T \mathbf{W}_x \mathbf{y}$, we have for $k = 0, 1, \ldots,$

$$\begin{aligned} \hat{\beta}_k &= \mathbf{e}_{k+1}^T \hat{\boldsymbol{\beta}} = \mathbf{e}_{k+1}^T S_n^{-1} \mathbf{X}_x^T \mathbf{W}_x \mathbf{y} \\ &= \sum_{i=1}^n W_k^n \left(\frac{x_i - x}{h} \right) y_i, \end{aligned}$$

  where $S_n = \mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x$, and $W_k^n(t) = \mathbf{e}_{k+1}^T S_n^{-1} \{1, th, \ldots, (th)^p\}^T K(t)/h$.
- Like a kernel estimator, except now the kernel depends on design points and locations.
- Related to adaptive property to designs and boundary estimations.
- In exercise 2 you will show that

$$S_{n,j} = \sum_{i=1}^n K_h(x_i - x)(x_i - x)^j = nh^j f_x(x) \mu_j \{1 + o_P(1)\}.$$

- This implies $S_n = n f_x(x) H S H \{1 + o_P(1)\}$, where $H = \text{diag}(1, h, \ldots, h^p)$.
- Can then write

$$\hat{\beta}_k = \frac{1}{nh^{k+1} f(x)} \sum_{i=1}^n K_k^* \left(\frac{x_i - x}{h} \right) y_i \{1 + o_P(1)\},$$

  where $K_k^*(t) = \mathbf{e}_{k+1}^T S^{-1} (1, t, \ldots, t^p) K(t)$ is called the equivalent kernel.

Local polynomial regression - Equivalent kernel

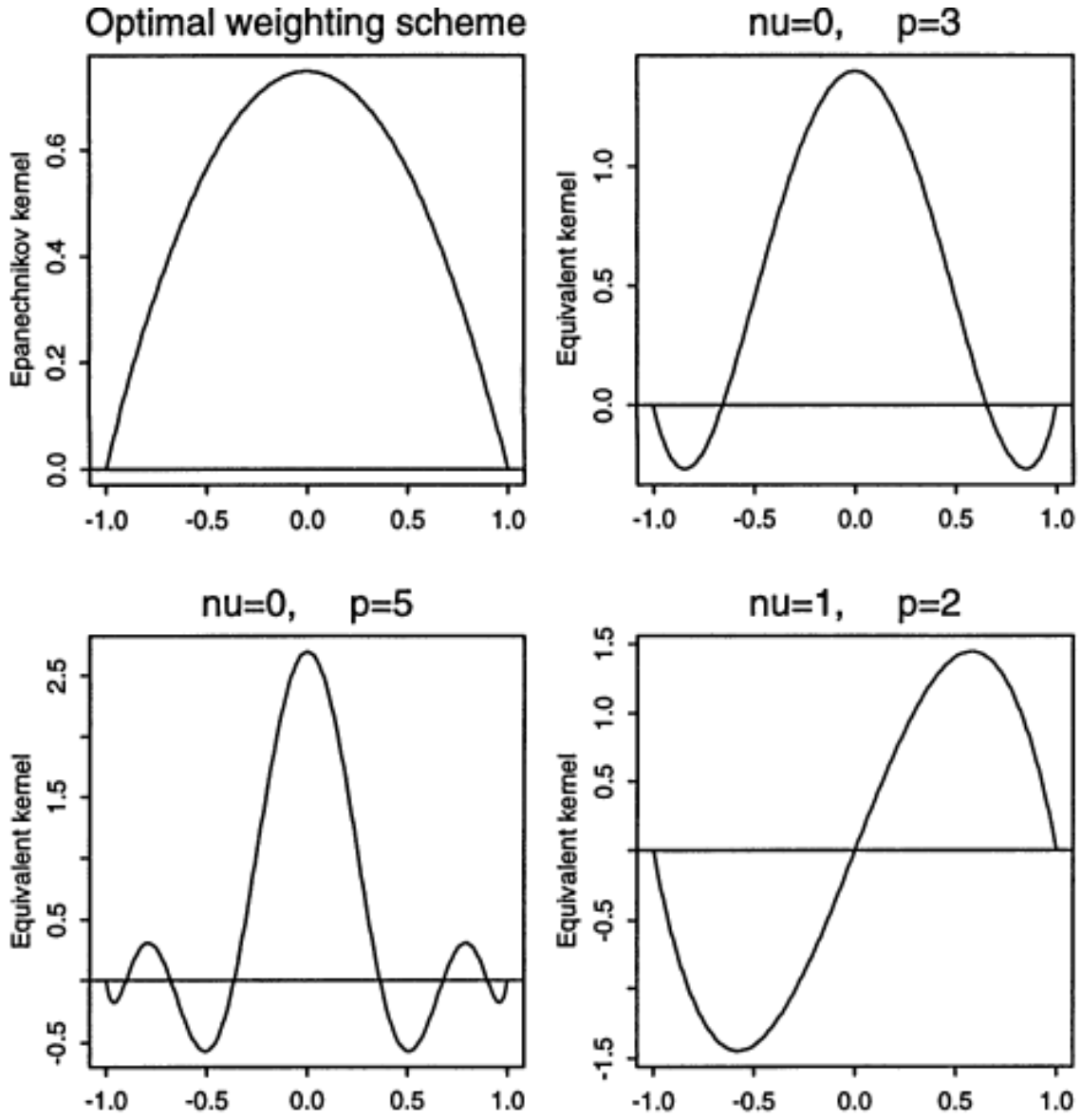


Figure: The Epanechnikov kernel and some corresponding equivalent kernels.

Local polynomial regression - choice of bandwidth

- Bandwidth can be constant or variable (locally or globally variable). We study optimal constant bandwidth only and locally variable bandwidth.
- A theoretical optimal bandwidth for estimating $f^{(k)}(x)$ is obtained by minimizing the conditional Mean Square Error (MSE) given by

$$\text{Bias}\{\widehat{f^{(k)}}(x)|\mathbb{X}\}^2 + \text{var}\{\widehat{f^{(k)}}(x)|\mathbb{X}\}.$$

- This leads to

$$h_{\text{opt}}(x) = C_{k,p}(K)\left[\frac{\sigma^2(x)}{\{f^{(p+1)}(x)\}^2 f_x(x)}\right]^{1/(2p+3)} n^{-1/(2p+3)},$$

where

$$C_{k,p}(K) = \left[\frac{(p+1)!(2k+1)\int K_k^*(t)dt}{2(p-k+1)\{\int t^{p+1}K_k^*(t)dt\}^2}\right]^{1/(2p+3)}.$$

- The order $h_{opt}(x) = O(n^{-1/(2p+3)})$ is not difficult to obtain, despite the complicated expressions above.
- For a constant bandwidth, we minimize the conditional weighted MISE (Mean Integrated Squared Error)

$$\int ([\text{Bias}\{\widehat{f^{(k)}}(x)|\mathbb{X}\}]^2 + \text{var}\{\widehat{f^{(k)}}(x)|\mathbb{X}\})w(x)dx.$$

The result is still $h_{\text{opt}} = O(n^{-1/(2p+3)})$.

Local polynomial regression - choice of bandwidth

- Practically, there are a lot of ways and papers contributed to choosing bandwidth.
- Cross-validation or GCV is one option. Construction of AIC or BIC criterion is another.
- Will not cover in this course.

Local polynomial regression - automatic carpentry

- A point x is a boundary point if $x \pm h$ lies outside the design region (corr. to a kernel K with bounded support $[-1, 1]$).
- It causes huge problems for most smoothing techniques, including large boundary bias. Boundary correction is usually needed for smoothing techniques.
- If $x = 1 - ch$ (assuming $x \in (-\infty, 1]$), then $c \geq 1$ corresponds to interior point, while $c < 1$ corresponds to boundary point.
- It turns out that the MSE is a continuous function of c when $p - k$ is odd, so that from boundary point to interior point, the risk changes continuously. Hence no boundary modification is needed. When $p - k$ is even, the risk is of higher order when c changes continuously to an interior point, and boundary modification is needed.

Local polynomial regression - automatic carpentry

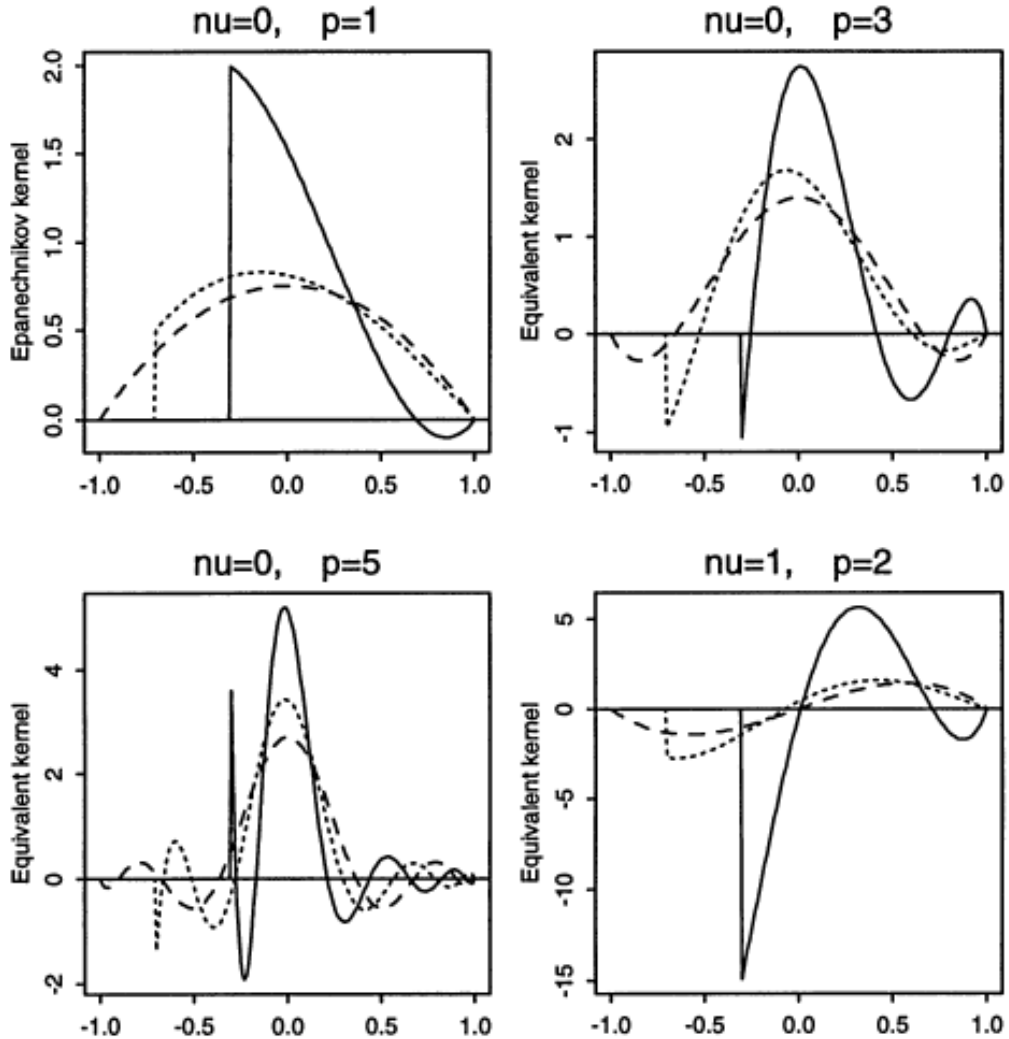


Figure: The Epanechnikov kernel and its equivalent kernels at the boundary points c=0.3 (solid line) and c=0.7 (dotted line) and interior points $c \geq 1$ (dashed line) for various values of p and ν