Lecture 12
Nonparametric Regression

Non Parametric Regression: Introduction

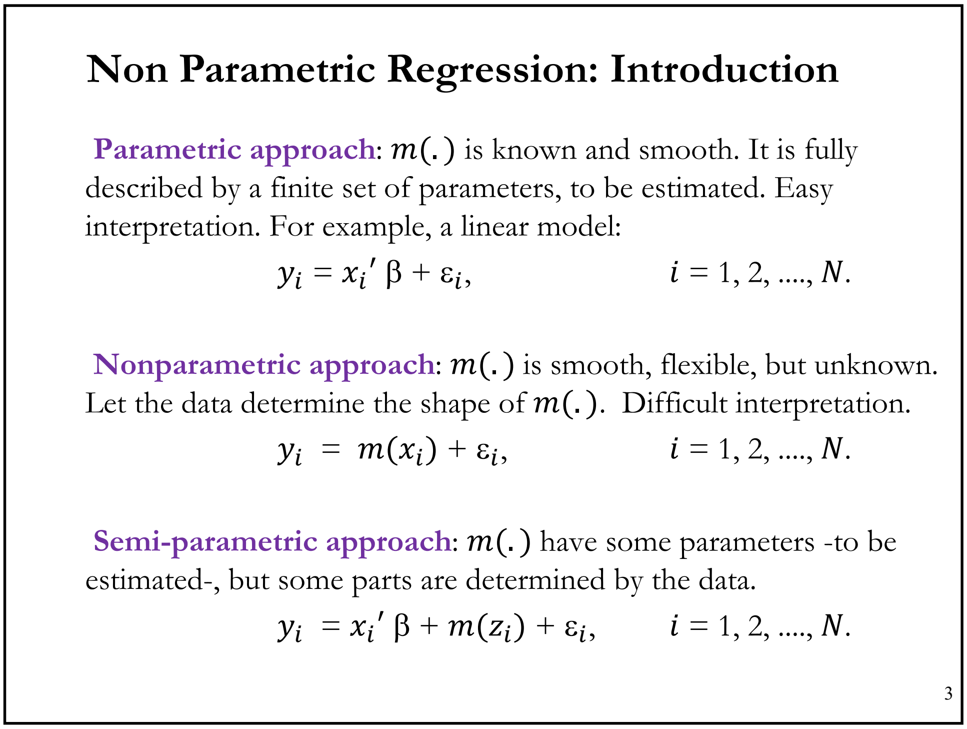
The goal of a regression analysis is to produce a reasonable analysis to the unknown response function f, where for N data points (x_i, y_i), the relationship can be modeled as

$$y_i = m(x_i) + \varepsilon_i, \qquad i = 1, 2, ..., N.$$

Note: $m(x_i) = \mathrm{E}[y_i \mid x_i]$ if $\mathrm{E}[\varepsilon_i \mid x_i] = 0$ –i.e., $\varepsilon \perp x$

We have different ways to model the conditional expectation function (CEF), $m(.)$:

- Parametric (single, global model).
- Semi-parametric (many local models).
- Nonparametric (all local, local data points have similar behavior)



Non Parametric Regression: Introduction

Parametric approach: $m(.)$ is known and smooth. It is fully described by a finite set of parameters, to be estimated. Easy interpretation. For example, a linear model:

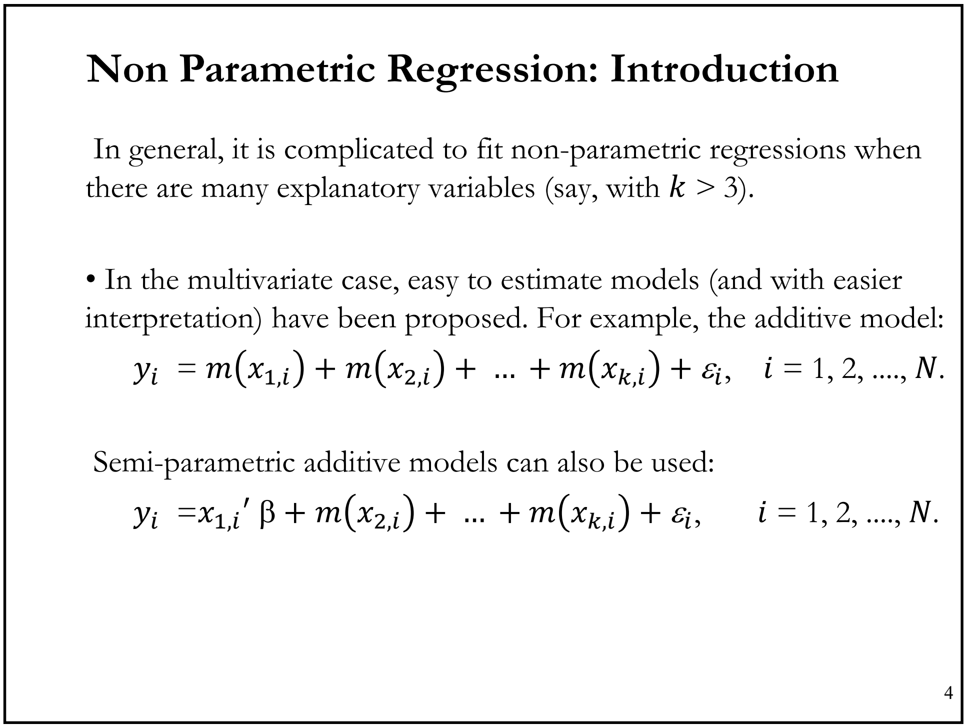
$$y_i = x_i{}'\,\beta + \varepsilon_i, \qquad i = 1, 2, ..., N.$$

Nonparametric approach: $m(.)$ is smooth, flexible, but unknown. Let the data determine the shape of $m(.)$. Difficult interpretation.

$$y_i \;=\; m(x_i) + \varepsilon_i, \qquad i = 1, 2, ..., N.$$

Semi-parametric approach: $m(.)$ have some parameters -to be estimated-, but some parts are determined by the data.

$$y_i \;= x_i{}'\,\beta + m(z_i) + \varepsilon_i, \qquad i = 1, 2, ..., N.$$



Non Parametric Regression: Introduction

In general, it is complicated to fit non-parametric regressions when there are many explanatory variables (say, with $k > 3$).

- In the multivariate case, easy to estimate models (and with easier interpretation) have been proposed. For example, the additive model:

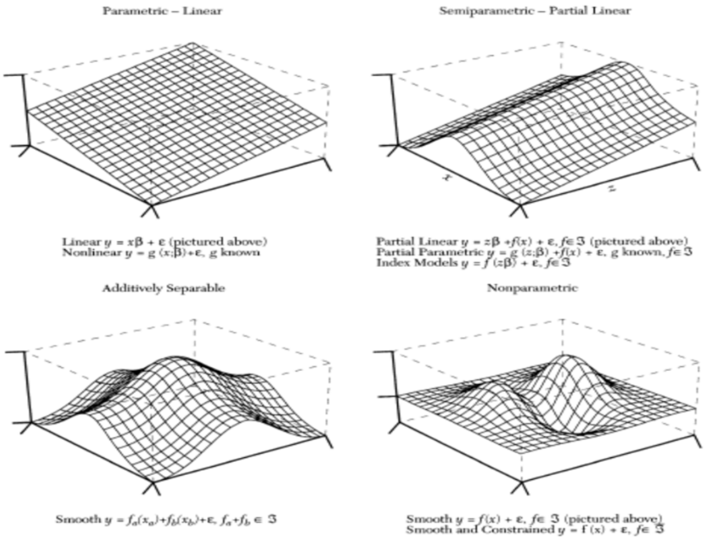
$$y_i \;= m(x_{1,i}) + m(x_{2,i}) + \;...\; + m(x_{k,i}) + \varepsilon_i, \quad i = 1, 2, ..., N.$$

Semi-parametric additive models can also be used:

$$y_i \;= x_{1,i}{}'\,\beta + m(x_{2,i}) + \;...\; + m(x_{k,i}) + \varepsilon_i, \qquad i = 1, 2, ..., N.$$

Non Parametric Regression: Introduction

Figure 2. Categorization of Regression Functions



Parametric – Linear

Linear $y = x\beta + \varepsilon$ (pictured above)
Nonlinear $y = g\,(x;\beta) + \varepsilon$, g known

Semiparametric – Partial Linear

Partial Linear $y = z\beta + f(x) + \varepsilon$, $f \in \mathfrak{I}$ (pictured above)
Partial Parametric $y = g\,(z;\beta) + f(x) + \varepsilon$, g known, $f \in \mathfrak{I}$
Index Models $y = f\,(z\beta) + \varepsilon$, $f \in \mathfrak{I}$

Additively Separable

Smooth $y = f_a(x_a) + f_b(x_b) + \varepsilon$, $f_a + f_b \in \mathfrak{I}$

Nonparametric

Smooth $y = f(x) + \varepsilon$, $f \in \mathfrak{I}$ (pictured above)
Smooth and Constrained $y = \mathrm{f}\,(x) + \varepsilon$, $f \in \bar{\mathfrak{I}}$

$\mathfrak{I}$ is a smooth family of functions. $\bar{\mathfrak{I}}$ is a smooth family with additional constraints such as monotonicity, concavity, symmetry or other constraints.

Regression: Smoothing

- We want to relate *y* with *x*, without assuming any functional form. First, we consider the one regressor case:

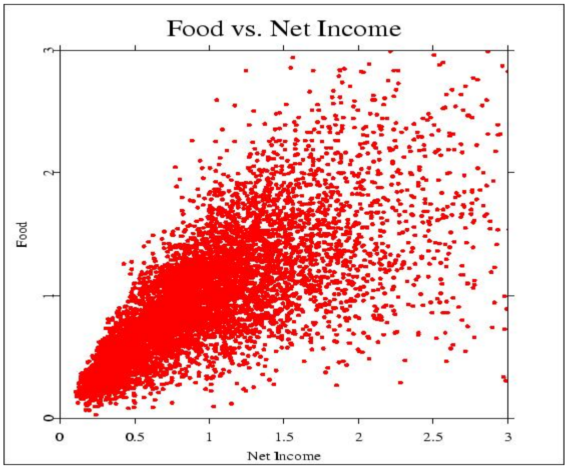
$$y_i \;=\; m(x_i) + \varepsilon_i, \qquad\qquad i = 1, 2, ..., N.$$

- In the CLM, a linear functional form is assumed: $m(x_i) \;=\; x_i{}'\,\boldsymbol{\beta}$.

- In many cases, it is not clear that the relation is linear.

- Non-parametric models attempt to discover the (approximate) relation between y_i and x_i. Very flexible approach, but we need to make some assumptions.

Regression: Smoothing



- The functional form between income and food is not clear from the scatter plot. From Hardle (1990).

Regression: Smoothing

- A reasonable approximation to the regression curve $m(x_i)$ will be the mean of response variables near a point x_i. This **local averaging** procedure can be defined as

$$\hat{m}_i(x) = \frac{1}{N}\sum_i^N w_i(x)\, y_i \qquad \text{-}w_i(x) = W_i(N, h, x_i)$$

- The averaging will *smooth* the data. The weights depend on the value of x and on a h. Recall that as h gets smaller, $\hat{m}(x)$ is less biased but also has greater variance.

Note: Every smoothing method to be described follows this form. Ideally, we give smaller weights for x's that are farther from x_i.

- It is common to call the regression estimator $\hat{m}(x)$ a **smoother** and the outcome of the smoothing procedure is called the **smooth**.

Regression: Smoothing – Example 1

- From Hansen (2013). To illustrate the concept, suppose we use the naive histogram estimator as the basis for the weight function, $w_i(x)$:

$$W_{N,h,i}(x_0) = \frac{I[(x_i - x_0) \le h]}{\sum_{i=1}^{N} I[(x_i - x_0) \le h]}$$

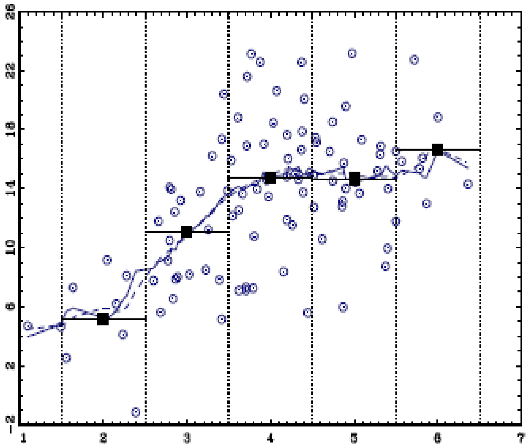
- Let x_0=2, h =0.5. The estimator $\widehat{m}(x = 2)$ is the average of the y_i for the observations such that x_i falls in the interval [1.5 ≤ x_i ≤ 2.5].

- Hansen simulates observations (see next Figure) and calculate $\widehat{m}(x)$ at x = 2, 3, 4, 5 & 6. For example, $\widehat{m}(x = 2)$ = 5.16, shown in the Figure as the first solid square.

- This process is equivalent to partitioning the support of x_i into the regions [1.5,2.5]; [2.5,3,5]; [3.5,4.5]; [4.5,5.5]; & [5.5,6.5]. It produces a step function. Reasonable behavior in the bins, but unrealistic jumps.

Regression: Smoothing – Example 1

- Figure 11.1 - Simulated data and $\widehat{m}(x)$ from Hansen (2013).



- Obviously, we can calculate $\widehat{m}(x)$ at a finer grid for x. It will track the data better. But, the unrealistic jumps (discontinuities) will remain.

Regression: Smoothing – Example 1

• The source of the discontinuity is the weights w_i are constructed from indicator functions, which are themselves discontinuous.

• If instead the weights are constructed from continuous functions, $K(.)$, $\hat{m}(x)$ will also be continuous in x. It will produce a true *smooth!* For example,

$$W_{N,h,i}(x_0) = \frac{K(\frac{x_i - x_0}{h})}{\sum_{i=1}^{N} K(\frac{x_i - x_0}{h})}$$

• The bandwidth h determines the degree of smoothing. A large $h$increases the width of the bins, increasing the smoothness of $\hat{m}(x)$. A small h decreases the width of the bins, producing a less smooth $\hat{m}(x)$.

Regression: Smoothing – Example 2

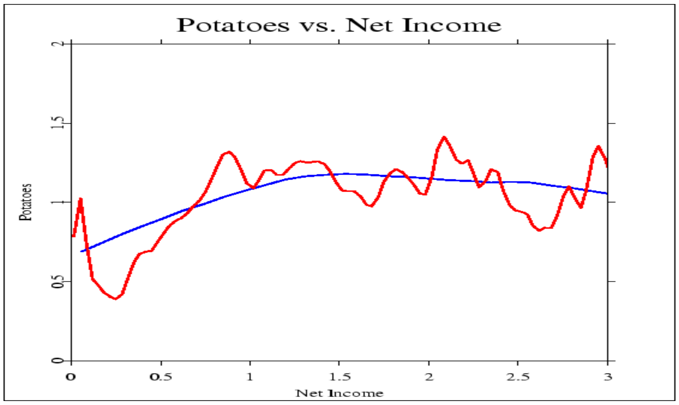
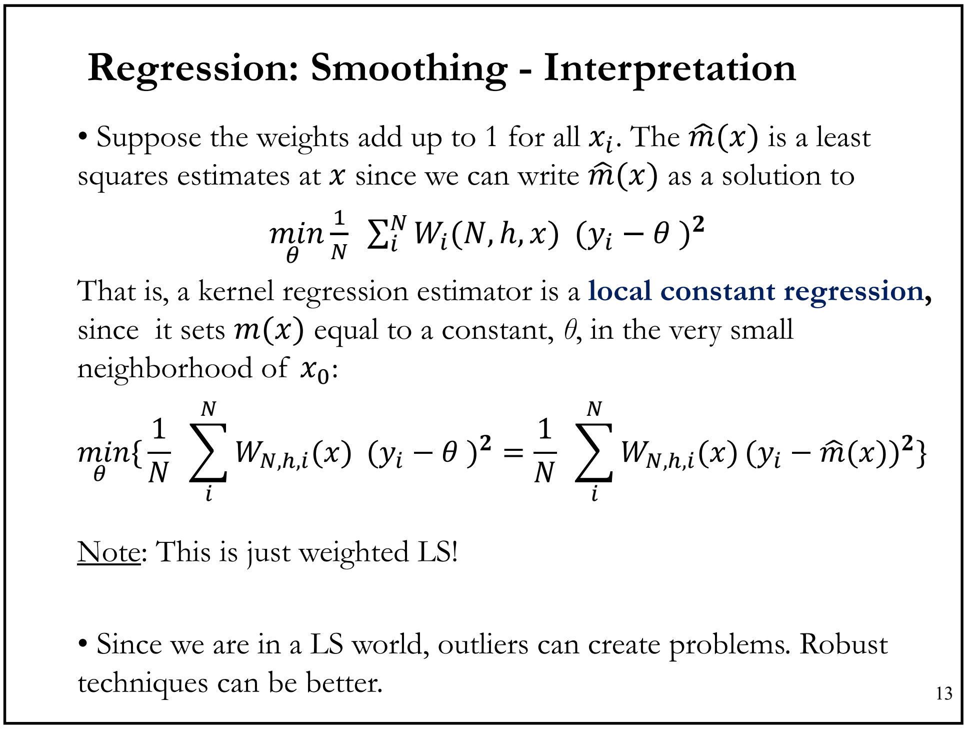


Figure 1. Expenditure of potatoes as a function of net income. h = 0.1, 1.0, N = 7125, year = 1973. Blue line is the *smooth*. From Hardle (1990).



Regression: Smoothing - Interpretation

- Suppose the weights add up to 1 for all x_i. The $\hat{m}(x)$ is a least squares estimates at x since we can write $\hat{m}(x)$ as a solution to

$$\min_{\theta} \frac{1}{N} \sum_i^N W_i(N, h, x)\ (y_i - \theta)^2$$

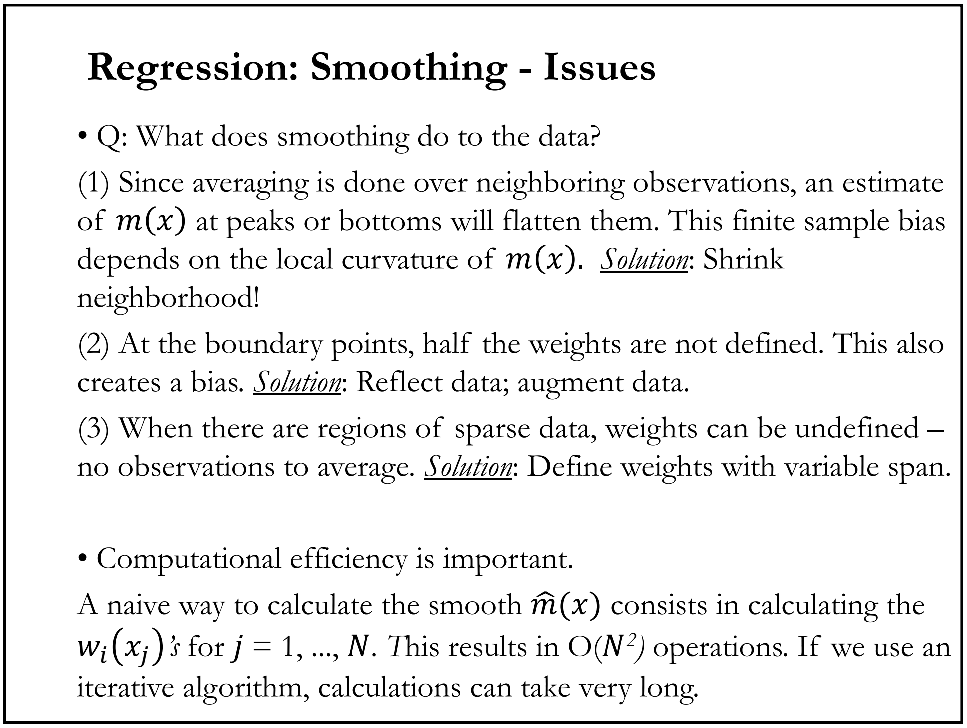
That is, a kernel regression estimator is a **local constant regression,** since it sets $m(x)$ equal to a constant, θ, in the very small neighborhood of x_0:

$$\min_{\theta}\{\frac{1}{N} \sum_i^N W_{N,h,i}(x)\ (y_i - \theta)^2 = \frac{1}{N} \sum_i^N W_{N,h,i}(x)\,(y_i - \hat{m}(x))^2\}$$

Note: This is just weighted LS!

- Since we are in a LS world, outliers can create problems. Robust techniques can be better.

13



Regression: Smoothing - Issues

- Q: What does smoothing do to the data?

(1) Since averaging is done over neighboring observations, an estimate of $m(x)$ at peaks or bottoms will flatten them. This finite sample bias depends on the local curvature of $m(x)$. *Solution*: Shrink neighborhood!

(2) At the boundary points, half the weights are not defined. This also creates a bias. *Solution*: Reflect data; augment data.

(3) When there are regions of sparse data, weights can be undefined – no observations to average. *Solution*: Define weights with variable span.

- Computational efficiency is important.

A naive way to calculate the smooth $\hat{m}(x)$ consists in calculating the $w_i(x_j)$'s for j = 1, ..., N. This results in O(N^2) operations. If we use an iterative algorithm, calculations can take very long.

Kernel Regression

• Kernel regressions are weighted average estimators that use kernel functions as weights.

• Recall that the kernel K is a continuous, bounded and symmetric real function which integrates to 1. The weight is defined by

$$W_{h,i}(x_0) = \frac{K_h(x_i - x_0)}{\hat{f}_h(x_0)}$$

where $\hat{f}_h(x_0) = \frac{1}{N} \sum_{i=1}^{N} K_h(x_i \; - \; x_0)$

$$K_h(u) = \frac{1}{h} \; K(\frac{u}{h})$$

• The functional form of the kernel virtually always implies that the weights are much larger for the observations where x_i is close to x_0. This makes sense!

Kernel Regression

Standard statistical formulas allow us to calculate E[$y|x$]:

$$\text{E}[y|x] = m(x) = \int y \, f(y|x) \, dy$$

where $f(y|x)$ is the distribution of y conditional on x. As always, we can express this conditional distribution in several ways. In particular:

$$\text{E}[y\,|\,x_i] = m(x) = \frac{\int y \; f(y,x) dy}{\int f(y,x) dy}$$

where the denominator is the marginal distribution $f(x)$.

Q: How can we estimate $m(x)$ using these formulas?

- First, consider first the denominator, $f(x)$. This is just the density of x. Estimate this using the KDE results from last class. For a given value of x (say, x_0) as:

$$\hat{f}_{Hist}(x_0) = \frac{1}{Nh} \sum_{i=1}^{N} K(\frac{x_i - x_0}{h})$$

Kernel Regression: NW Estimator

- First, consider first $f(x)$:

$$\hat{f}_{Hist}(x_0) = \frac{1}{Nh}\sum_{i=1}^{N} K(\frac{x_i - x_0}{h})$$

- Second, extending the KDE results suggests:

$$\int y\, f(y, x_0)dy = \frac{1}{Nh}\sum_{i=1}^{N} K(\frac{x_i - x_0}{h})\; y_i$$

• Plugging these two kernel estimates of the terms in the numerator and the denominator of the expression for $m(x)$ gives the **Nadaraya-Watson (NW) kernel estimator**, or **local constant** (**LC**) **estimator**:

$$\widehat{m}(x_0) = \frac{\sum_{i=1}^{N} K(\frac{x_i - x_0}{h})\; y_i}{\sum_{i=1}^{N} K(\frac{x_i - x_0}{h})}$$

• The NW estimator gives us a local average: We regress y_i locally, on a constant, weighting observations via their distance to x_0.

Kernel Regression: NW estimator - Different *K*

• The shape of the kernel weights is determined by K and the size of the weights is parameterized by h (h plays the usual smoothing role).

• The normalization of the weights with $\hat{f}_h(x_0) = \frac{1}{N}\sum_{i=1}^{N} K_h(\frac{x_i - x_0}{h})$ is called the **Rosenblatt-Parzen kernel density estimator**. It makes sure that the weights add up to 1.

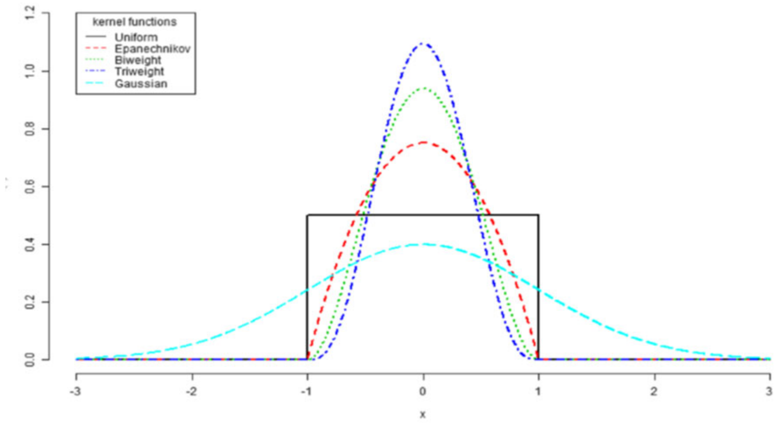
• Two important constants associated with a kernel function $K(.)$ are its variance $\sigma_K^2 = d_k$ and roughness R_k, (also denoted R(K)), which are defined as:

$$d_k = \int z^2 K(z)dz$$

$$R_k = \int (K(z))^2\, dz$$

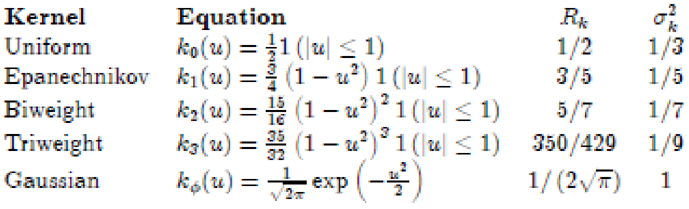
Kernel Regression: NW estimator - Different K

Many $K(.)$ are possible. Practical and theoretical considerations limit the choices. Usual choices: Epanechnikov, Gaussian, Quartic (biweight), and Tricube (triweight).



Kernel Regression: NW estimator - Different K

- Below we present the kernels functions, along with the roughness and variance for the most popular (second order) kernels: Epanechnikov, Gaussian, Quartic (biweight), and Tricube (triweight).



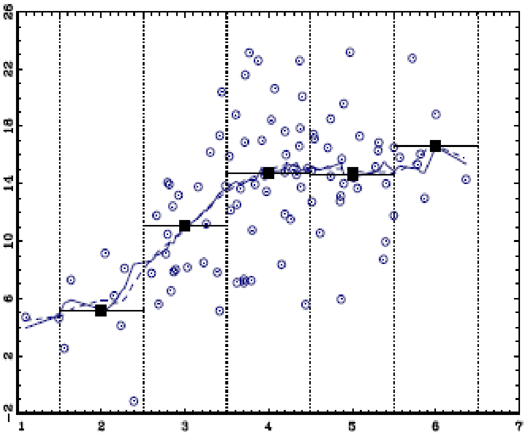
Kernel	Equation	R_k	σ_k^2
Uniform	$k_0(u) = \frac{1}{2}1\left(\lvert u\rvert \le 1\right)$	$1/2$	$1/3$
Epanechnikov	$k_1(u) = \frac{3}{4}\left(1-u^2\right)1\left(\lvert u\rvert \le 1\right)$	$3/5$	$1/5$
Biweight	$k_2(u) = \frac{15}{16}\left(1-u^2\right)^2 1\left(\lvert u\rvert \le 1\right)$	$5/7$	$1/7$
Triweight	$k_3(u) = \frac{35}{32}\left(1-u^2\right)^3 1\left(\lvert u\rvert \le 1\right)$	$350/429$	$1/9$
Gaussian	$k_\phi(u) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{u^2}{2}\right)$	$1/\left(2\sqrt{\pi}\right)$	1

- Figure 11.1 shows the NW estimator with Epanechnikov kernel and h = 0.5 with the dashed line. (The full line uses a uniform kernel.)

- The Epanechnikov kernel enjoys optimal properties and, in general, is the preferred kernel in statistics.

Kernel Regression: NW estimator – Example 1

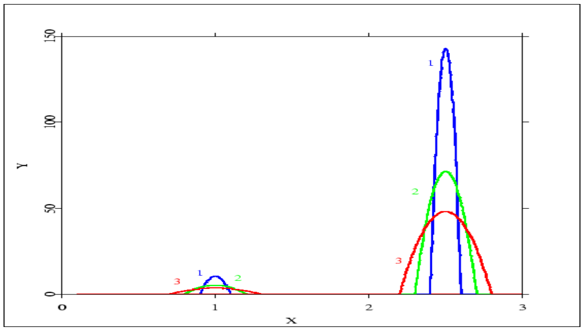
- Figure 11.1 - Simulated data and $\hat{m}(x)$ from Hansen (2013).



- Obviously, we calculated $\hat{m}(x)$ before with a constant in each interval. Now, we estimate $\hat{m}(x)$ under NW (dashed line) .

Kernel Regression: Epanechnikov kernel.

Figure 3. The effective kernel weights for the food/ income data: At x = 1 and x = 2.5 for h = 0.1 (label 1, blue), h = 0.2 (label 2, green), h = 0.3 (label 3, red) with *Epanechnikov kernel.* From Hardle (1990).



- The smaller h*,* the more concentrated the w_i's. In sparse regions, say x = 2.5 (low marginal pdf), it gives more weight to observations around x.

Kernel Regression: NW estimator – Properties

- The NW estimator is defined by

$$\hat{m}_h(x_0) = \frac{\sum_{i=1}^{N} K_h(x_i - x_0)\ y_i}{\sum_{i=1}^{N} K_h(x_i - x_0)} = \sum_{i=1}^{N} w_{N,h,i}(x_0)\ y_i$$

- Similar situation as in KDE: No finite sample distribution theory for $\hat{m}_h(x)$. All statistical properties are based on asymptotic theory.

- Details. One regressor $(d = 1)$, but straightforward to generalize.
Fix x. Note that $y_i = m(x_i) + \varepsilon_i = m(x) + [m(x_i) - m(x)] + \varepsilon_i$
Then,

$$\frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x_i - x}{h}\right) y_i =$$

$$= \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x_i - x}{h}\right) * (m(x) + [m(x_i) - m(x)] + \varepsilon_i)$$

$$= \hat{f}(x)m(x) + \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x_i - x}{h}\right)(m(x_i) - m(x)) + \frac{1}{Nh}K\left(\frac{x_i - x}{h}\right)\varepsilon_i$$

Kernel Regression: NW estimator – Properties

It follows that

$$\hat{m}(x) = m(x) + \hat{m}_1(x)/\hat{f}(x) + \hat{m}_2(x)/\ \hat{f}(x)$$

(1) $\boldsymbol{\hat{m}_2(x)}$= $\frac{1}{Nh}K\left(\frac{x_i - x}{h}\right)\varepsilon_i$

- **Mean**.

Since E[$\varepsilon_i|x_i$] = 0 ⇒ E[$\hat{m}_2(x)$] = 0.

- **Variance**.

$$Var[\hat{m}_2(x)] = \frac{1}{Nh^2}E[K(\frac{x_i - x}{h})\varepsilon_i]^2 = \frac{1}{Nh^2}E[K(\frac{x_i - x}{h})^2\sigma^2(\boldsymbol{x}_i)]$$

(by conditioning), and then

$$Var[\hat{m}_2(x)] = \frac{1}{Nh^2}\int K(\frac{z - x}{h})^2\sigma^2(z)f(z)\ dz$$

Next, change of variables, $\frac{z - x}{h} = u; h\ du = dz$:

Kernel Regression: NW estimator – Properties

- **Variance**.

Change of variables, $\frac{z-x}{h} = u$. Assume $\sigma^2(x)$and $f(x)$ are smooth:

$$Var[\hat{m}_2(x)] = \frac{1}{Nh^2}\int K(u)^2\sigma^2(x+hu)f(x+hu)\ (hdu)$$

$$= \frac{1}{Nh}\int K(u)^2\sigma^2(x)f(x)\ du + o(\frac{1}{Nh})$$

$$= \frac{\sigma^2(x)f(x)}{Nh}\ R(K) + o(\frac{1}{Nh})$$

- **Distribution**

We can apply the CLT to obtain that as $h \to 0$, and $Nh \to \infty$:

$$\sqrt{Nh}\ \hat{m}_2(x)\xrightarrow{d}N(0,\sigma^2(x)f(x)R(K))$$

Kernel Regression: NW estimator – Properties

(2) $\hat{m}_1(x)$ $= \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x_i - x}{h}\right)(m(x_i) - m(x))$

- Mean

$$E[\hat{m}_1(x)] = \frac{1}{h}\ E\left[\sum_{i=1}^{N} K\left(\frac{x_i - x}{h}\right)\left(m(x_i) - m(x)\right)\right] =$$

$$= \int K(u)\ \left(m(x+hu) - m(x)\right)f(x+hu)\ du$$

Expand $m(x+hu)$ and $f(x+hu)$ into (2nd- and 1st-order, respectively) Taylor expansions around x: Up to $o(h^2)$ we get:

$$E[\hat{m}_1(x)] = \int K(u)\ \left(m(x+hu) - m(x)\right)f(x+hu)\ du$$

$$= \int K(u)\ (m(x) + hum'(x) + \frac{(hu)^2}{2}m''(x)\ - m(x))\ * (f(x) + hu\ f'(x))du$$

$$\approx hm'(x)f(x)\int K(u)\ u\ du + h^2\{\frac{m''(x)f(x)}{2} + m'(x)f'(x)\}\int K(u)u^2\ du$$

$$\approx hm'(x)f(x)\ \kappa_1 + h^2\ \{\frac{m''(x)f(x)}{2} + m'(x)f'(x)\}\ \kappa_2 = h^2\ B(x)\ \kappa_2\ f(x)$$

Kernel Regression: NW estimator – Properties

- Then, we get

$$E[\hat{m}_1(x)] = h^2\,\kappa_2\,B(x)f(x)$$

where $B(x) = \{\frac{m''(x)f(x)}{2} + m'(x)f'(x)\}\,/f(x)$

- **Variance**. A similar expansion shows that Var[$\hat{m}_1(x)$] is O($\frac{h^2}{Nh}$), which is of smaller order than O($\frac{1}{Nh}$).

Thus, as $h \to 0$, & $Nh \to \infty$, $\sqrt{Nh}\,[\hat{m}_1(x) - h^2\,\kappa_2\,B(x)f(x)] \xrightarrow{p} 0$

and since $\hat{f}(x) \xrightarrow{p} f(x) \Rightarrow \quad \sqrt{Nh}[\frac{\hat{m}_1(x)}{\hat{f}(x)} - h^2\,\kappa_2\,B(x)] \xrightarrow{p} 0$

Kernel Regression: NW estimator – Properties

- This bias is of size O(h^2). Intuitively, the bias is larger the "curvier" $m(x_0)$ is -i.e., the larger $m'(x_0)$ and $m''(x_0)$ are.

The kernel regression estimator, $\hat{m}(x)$, is *consistent*. But, convergence is at the rate sqrt(Nh), not the usual sqrt(N).

- By CLT, we get under general assumptions, *asymptotically normality:*

$$\sqrt{Nh}\,[\hat{m}(x) - m(x) - h^2\,\kappa_2\,B(x)] \xrightarrow{d} N(0,\ \frac{\sigma^2(x)R(K))}{f(x)})$$

- The MSE = variance + bias^2. Given our asymptotic results, we can get the AMSE[$\hat{m}(x)$]:

$$\text{AMSE}[\hat{m}(x)] \approx \frac{1}{Nh}\,\frac{\sigma^2(x)R(K))}{f(x)} + [h^2\,\kappa_2\,B(x)]^2$$

Kernel Regression: NW estimator – Properties

- Notes about asymptotic distribution:

- The asymptotic distribution depends on the kernel through $R(K)$ – the roughness- and κ_2 –the 2nd moment of z.

- The optimal kernel minimizes R_k; the same as for density estimation. Therefore, the Epanechnikov family is optimal for regression.

- The optimal h depends on the first and second derivatives of $m(x)$, not on $f(x)$

- Rules of thumb for h designed for $f(x)$ have no justification.

Kernel Regression: NW estimator – C.I.s

Given the asymptotic normality, it is easy to construct C.I.'s.

Usual steps:

1) Compute $\widehat{m}(x)$, and, using kernel density estimation, $\hat{f}(x)$.

2) Estimate $\sigma^2(x)$. R_k, the roughness, can be obtained from Tables.

3) Select α% level and use usual formula.

Note that we are not estimating the bias:

$$Bias[\widehat{m}(x)] - h^2\, \kappa_2 B(x)\, f(x)$$

where $B(x) = \{\frac{m''(x)f(x)}{2} + m'(x)f'(x)\}\, / f(x)$

It is complicated, since it needs estimates of derivatives. In general, it adds noise to the C.I. That is, we do not estimate an asymptotic exact C.I.

Kernel Regression: NW estimator – C.I.s

C.I.'s tend to be wider at the boundaries and when the data is sparse.

Even if we compute the bias, asymptotic C.I.'s are an approximation. A bootstrap may work better.

Kernel Regression: NW estimator – CAPM

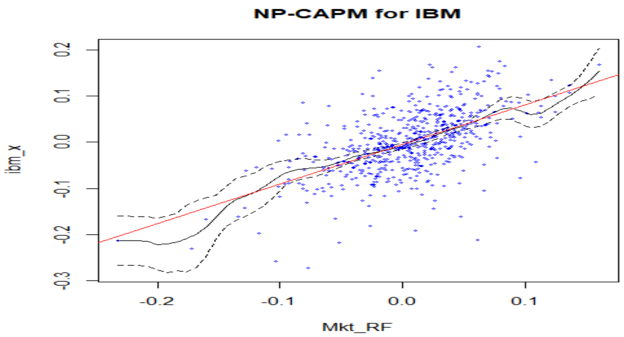
Example: We use NW to estimate a CAPM like non-parametric relation between IBM excess returns (ibm_x) and Market excess returns (Mkt_RF). We use the **np** R package.

```
SFX_da <- read.csv("http://www.bauer.uh.edu/rsusmel/4397/Stocks_FX_1973.csv",
 head=TRUE, sep=",")
## Extract variables from imported data
x_ibm <- SFX_da$IBM                          # extract IBM price data
x_Mkt_RF <- SFX_da$Mkt_RF                    # extract Market excess returns (in %)
x_RF <- SFX_da$RF                            # extract Risk-free rate (in %)
# Define log returns & adjust size of variables accordingly
T <- length(x_ibm)                           # sample size
lr_ibm <- log(x_ibm[-1]/x_ibm[-T])           # IBM log returns (in decimal returns)
Mkt_RF <- x_Mkt_RF[-1]/100                   # Adjust sample size to ( T-1)
RF <- x_RF[-1]/100                           # Adjust sample size & use decimal returns.
ibm_x <- lr_ibm - RF
library(np)
bw <- npregbw(formula = ibm_x ~ Mkt_RF)    # pick h with LS CV bandwidths (NW is default)
```

Kernel Regression: NW estimator – CAPM

Example (continuation):

```
model <- npreg(bws = bw, gradients = TRUE)
summary(model)
capm_ibm <- lm(ibm_x ~ Mkt_RF)

plot(bw, plot.errors.method="bootstrap", main = "NP-CAPM for IBM")
points(Mkt_RF, ibm_x, cex=.4, col="blue")
abline(capm_ibm, col="red")
```



h = 0.0129

Kernel Regression: NW estimator – CAPM

(1) Applied to truly linear data, the NW estimator can be poor.

- Let d = 1 and the true conditional mean is linear $y_i = x_i{}'\,\beta$, with no error. The behavior of the NW estimator depends on the marginal distribution of X.

- If they are not spaced at uniform distances, then $\hat{m}(x) \neq m(x)$. The NW estimator applied to purely linear data yields a nonlinear output.

- The choice of h may not help. As h increases, the estimator becomes a constant, not a linear function.

(2) Poor behavior at the boundaries of X. Suppose $m(x)$ is positively sloped, at the right boundary, the NW estimator will be upward biased. In fact, the estimator is inconsistent at the boundary.

- This restricts application of the NW estimator to interior points.

Kernel Regression: Derivatives

• The same principles behind kernel estimation can be used to estimates the derivatives of the regression function. These derivatives can be used to estimate partial effects.

• If the weights are sufficiently smooth and h is properly chosen, the derivative estimator is consistent.

• Taking the k-*th* derivative of $\widehat{m}(x)$:

$$\hat{m}^{(k)}(x_0) = N^{-1}h^{-(k+1)}\sum_{i=1}^{n} w^{(k)}(\frac{x_i - x_0}{h})y_i$$

• The kernel estimate of the *k-th* derivative is also a local average.

Kernel Regression: Local Linear Estimator

• We motivated the NW estimator at x as an average of the y_i's for observations in a neighborhood of x: A local constant approximation.

• Instead, we can do OLS in the same neighborhood. If we use a weighting function, this is called the **local linear** (**LL**) **estimator**.

• The idea is to fit the local linear model

$$y_i = \alpha + (x_i - x)'\,\beta + \varepsilon_i,$$

• We use $(x_i - x)$ rather than x_i to have $m(x_i)$ = E[$y_i\,|\,x_i = x$] = α.

• We do OLS with observations such that $|x_i - x| \leq h$. That is,

$$\min_{\alpha,\beta} \frac{1}{N}\ \sum_i^N (y_i - \alpha +\ (x_i - x)'\,\beta)^2\ I[|x_i - x| \leq h]$$

Kernel Regression: Local Linear Estimator

- We have a (locally) weighted regression of y_i on x_i and a constant, with the indicator (uniform) function as a weight function. Then, the LL LS estimator is:

$$\hat{\delta}(x) = (\textstyle\sum_i^N I[\,|x_i - x|\, \leq h]\, \boldsymbol{Z}_i{}'\boldsymbol{Z}_i)^{-1}\ (\textstyle\sum_i^N I[|x_i - x| \leq h]\, Z_i{}'y_i)$$

where $\delta = (\alpha, \beta)'$, and $\boldsymbol{Z}_i$ = [1 $(x_i - x)$]'.

- We have a weighted LS (WLS) problem, which can be generalized to:

$$\min_{\alpha,\beta} \frac{1}{N} \textstyle\sum_i^N W_i(N,h,x)\ \ (y_i - \alpha +\ (x_i - x)'\,\beta)^{\mathbf{2}}$$

where $W_i(N,h,x)$ =K(). A higher order polynomial can also be used.

- We get different (α, β) at different x_0.

Kernel Regression: LL Estimator – WLS

- Using matrix notation:

$$\min_{\delta} (y - \boldsymbol{Z}\delta)'\, K(\boldsymbol{Z})(y - \boldsymbol{Z}\delta)$$

where $\delta = (\alpha, \beta)'$, $\boldsymbol{Z}_i$ = [1 $(x_i - x)$]', and $K(\boldsymbol{Z})$ is an NxN diagonal matrix with i-th element $K(\frac{x_i - x}{h})$.

- Then:

$$\hat{\delta}(x) = \begin{bmatrix} \hat{\alpha}(x) \\ \hat{\beta}(x) \end{bmatrix} = (\textstyle\sum_i^N K(\frac{x_i-x}{h})\, Z_i{}'Z_i)^{-1}\ (\textstyle\sum_i^N K(\frac{x_i-x}{h})\, Z_i{}'y_i)$$
$$= (\boldsymbol{Z}'K(\boldsymbol{Z})\boldsymbol{Z})^{-1}\, \boldsymbol{Z}'K(\boldsymbol{Z})y$$

- We have a (locally) WLS of y_i on x_i, with the kernel providing weights.

Note: OLS is just a special case of LL estimation, we set $K(\boldsymbol{Z}) = \boldsymbol{I}$.

Kernel Regression: LL Estimator – WLS

- LL estimator is more popular than the LC (NW) estimator since it preserves linear data & behaves better at the boundaries.

- In principle, we can add higher order polynomial terms, which would make it easier to take higher order derivatives. In these cases, we get a **Local-polynomial (LP) LS estimator**.

For example, with a second-order polynomial,

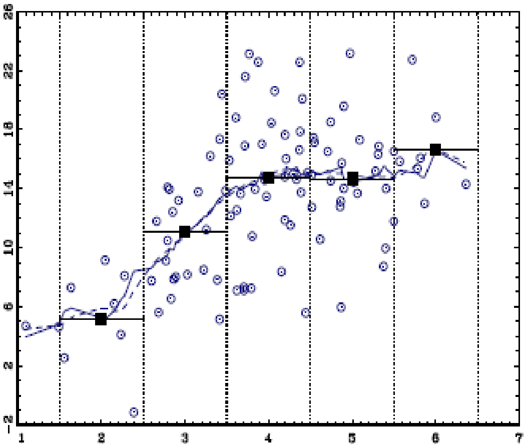
$$y_i = \alpha + (x_i - x)\ \beta_1 + (x_i - x)^2\ \beta_2 + \varepsilon_i,$$

which, by defining $\boldsymbol{Z}_i$ = [1 $(x_i - x)$ $(x_i - x)^2$], can be again easily estimated, locally, with WLS:

$$\hat{\delta}(x) = (\boldsymbol{Z}'K(\boldsymbol{Z})\boldsymbol{Z})^{-1}\ \boldsymbol{Z}'K(\boldsymbol{Z})y$$

Kernel Regression: LL Estimator – Example

- Figure 11.1 - Simulated data and $\hat{m}(x)$ from Hansen (2013).



- $\hat{m}(x)$ estimated under NW (dashed line) and LL (points). Overall, very similar smooths.

Kernel Regression: LL Estimator – LOWESS

- A popular local regression estimator is ***lo**cally* ***we**ighted* ***s**catterplot* ***s**moothing* (**lowess** or just **loess**) introduced by Cleveland (1979).

- It uses a smoothing variable h, a distance function from x_0 to x_k (its k-th NN), and weights given by a **tricubic kernel**:

$$K(z_i) = 1 * (1 - |z_i|^3)^3 \; I[|z_i| < 1].$$

where z_i is a ratio, determined by $|x_0 - x_i\,|/d_k$, where d_k is the maximum distance from point x_i to the farthest point in the neighborhood considered. The weights vary from 1 at the smoothing point where $x_i = x_0$ to 0 at the point x_i = furthest in the neighborhood considered.

- In principle, we work with a λ-degree polynomial, where λ is usually set as 1 or 2.

Kernel Regression: LL Estimator – LOWESS

- A popular local regression estimator is ***lo**cally* ***we**ighted* ***s**catterplot* ***s**moothing* (**lowess** or just **loess**) introduced by Cleveland (1979).

- It uses a smoothing variable h, a distance function around x_0, and weights given by a **tricubic kernel**:

$$K(z_i) = 1 * (1 - |z_i|^3)^3 \; I[|z_i| < 1].$$

where z_i is a ratio, given by $|x_0 - x_i\,|/d_q$, where $d_q(x)$ is computed such that the proportion of x_i's values within x_0 is q. That is,

$$d_q(x) = \min\{d > 0 : \tfrac{1}{N}\textstyle\sum_i^N I[|x_i - x| \leq d] < q\}$$

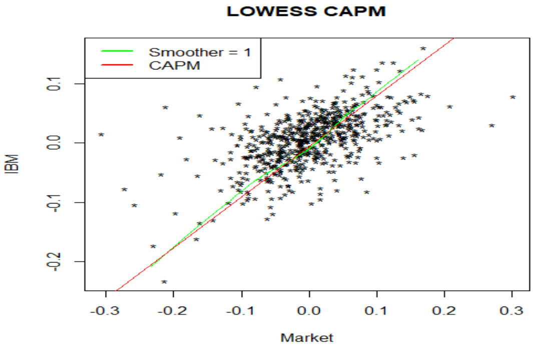
A usual choice for q = 0.5.

- In principle, we can work with a p-degree polynomial; in general p is usually set as 1 or 2.

Kernel Regression: LL Estimator – Example 1

Example: We use LOWESS to estimate the relation between (ibm_x) & (Mkt_RF). We use R command lowess (default h = 1).

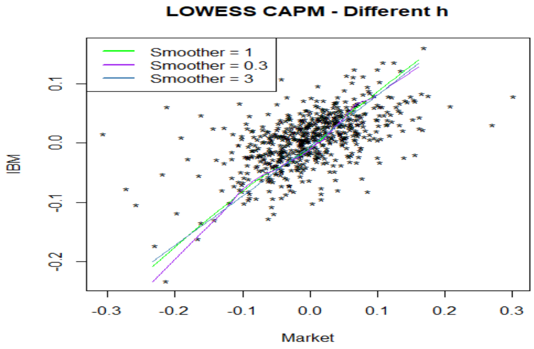
```
plot(ibm_x, Mkt_RF, xlab = "Market", ylab = "IBM", pch ="*",  main = "LOWESS CAPM")
low_fit <-lowess(ibm_x ~ Mkt_RF)
lines(lowess(ibm_x ~ Mkt_RF), col="green")
abline(capm_ibm, col="red")
```



Kernel Regression: LL Estimator – Example 1

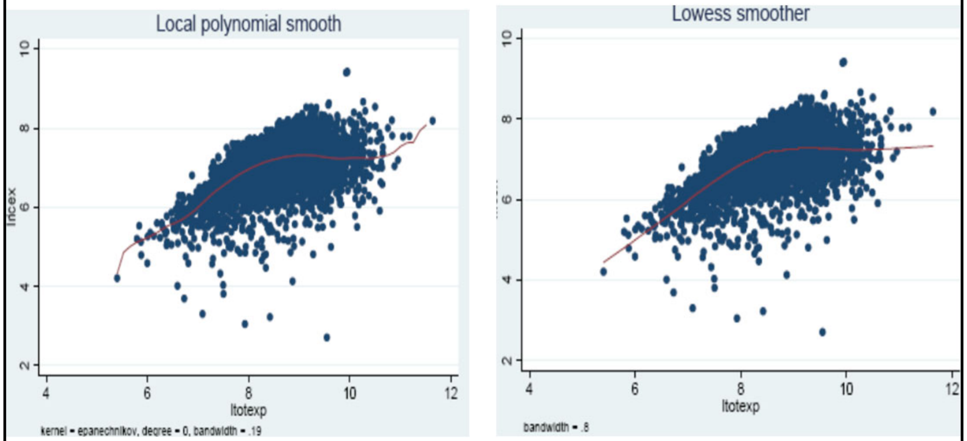
Example (continuation):

```
plot(ibm_x, Mkt_RF, xlab = "Market", ylab = "IBM", pch ="*",  main = "LOWESS CAPM -
 Different h")lines(lowess(ibm_x ~ Mkt_RF), col="green")
lines(lowess(ibm_x ~ Mkt_RF, f = 0.3), col='purple')                  # f = h = span
lines(lowess(ibm_x ~ Mkt_RF, f = 3), col='steelblue')
legend('topleft',       col = c('green', 'purple', 'steelblue'),
lwd = 2,      c('Smoother = 1', 'Smoother = 0.3', 'Smoother = 3'))
```



Kernel Regression: LL Estimator – Example 2

Rice expenditures as a total of Log total expenditures.



Kernel Regression: NW or LL Estimator?

• In contrast to the NW estimator, the LL estimator preserves linearity in the data. That is, if the true data is linear, for any sub-sample, a local linear regression fits exactly, so $\widehat{m}(x_0) = m(x_0)$.

As $h \rightarrow \infty$, the LL estimator collapses to OLS of y_i on x_i. That is, we can think of LL as a nonparametric generalization of OLS.

• The asymptotic distribution of the LL estimator is similar to that of the NW estimator. The bias term is simpler, the $m'(x_0)$ and $f'(x_0)$ disappear. The asymptotic variance is the same.

Q: If LL improves on NW, why not use a local polynomial of order *p?* It is possible and doable. In practice, when $d > 1$, applying polynomial methods is not easy.

Kernel Regression: NW or LL Estimator?

• Strictly speaking, we cannot rank the AMSE of the NW versus the LL estimator.

• The AMSE of the LL estimator only depends on $m''(x_0)$; while that of the NW estimator also depends on $m'(x_0)$. We expect this to translate into reduced bias.

• Since both estimators have the same asymptotic variance, the statistics literature prefers the LL estimator.

• According to Bruce Hansen (2013), caution is warranted. In simple simulations, the LL estimator does not always beat the NW estimator.

Kernel Regression: NW or LL Estimator?

• Hansen's interesting findings:

- When the regression function $m(x_0)$ is quite flat, the NW estimator does better. When the regression function is steeper and curvier, the LL estimator tends to do better.

- Intuition from above result: In finite samples the NW estimator tends to have a smaller variance. An advantage in contexts where estimation bias is low (such as when the regression function is flat).

Note: In many economic contexts, it is believed that the regression function may be quite flat with respect to many regressors. In this context it may be better to use NW rather than LL.

Kernel Regression: Weighted NW Estimator?

- Hall et al. (1999) proposed a weighted *NW estimator* as defined by

$$\hat{m}_h(x_0) = \frac{\sum_{i=1}^{N} p_i(x_0) K_h(x_i - x_0)\ y_i}{\sum_{i=1}^{N} p_i(x_0) K_h(x_i - x_0)} = \sum_{i=1}^{N} w_{N,h,i}(x_0)\ y_i$$

where $p_i(x)$ are weights. The weights satisfy:

$p_i(x) \geq 0$

$\sum_i^N p_i(x) = 1.$

$\sum_i^N p_i(x)\ K_h(x_i - x)\ (x_i - x) = 1.$

- The first two requirements define $p_i(x)$ as weights. The third equality requires the weights to force the kernel function to satisfy local linearity.

Kernel Regression: Weighted NW Estimator?

- The weights are determined by empirical likelihood. Specifically, for each x; you maximize $\sum_i^N ln[p_i(x)]$ s.t. the above constraints.

The solutions take the form

$$p_i(x) = \frac{1}{n\left(1 + \lambda'(X_i - x) K\left(H^{-1}(X_i - x)\right)\right)}$$

where λ is a LM, found by numerical optimization.

- The estimator $\hat{m}(x)$ has the same asymptotic distribution as the LL estimator.

Kernel Regression: Weighted NW Estimator?

- When $y_i \geq 0$; the standard and weighted NW estimators also satisfy $\hat{m}(x) \geq 0$. This is good ($m(x)$ is non-negative!). On the other side, the LL estimator is not necessarily non-negative.

Disadvantage: More computationally intensive than the LL estimator. The EL weights must be found separately for each x_0 at which $\hat{m}(x_0)$ is calculated.

Kernel Regression: Residuals, Fit & CV

- We are used to use the fitted residuals to construct GOF measures. The residuals are defined as usual:

$$e_i = y_i - \hat{m}(x_i), \qquad i = 1, 2, ..., N.$$

Problem: In general, but especially when h is small, it is hard to view e_i as a GOF measure. As $h \to 0$, $\hat{m}(.) \to y_i$ (and $e_i \to 0$). This indicates *overfitting* as the true error is not zero.

Solution: Measure the fit of the regression at $x = x_i$ by re-estimating the model excluding the i-th observation (notation: "$-i$," the i-th observation excluded). This is the **leave-one-out** estimation For NW regression, we get:

$$\hat{m}_{-i}(x) = \frac{\sum_{j\neq i}^{N} K_h(x - x_j)\ y_j}{\sum_{j\neq i}^{N} K_h(x - x_j)} = \sum_{j\neq i}^{N} w_{N,h,-i}(x)\ \ y_j$$

Kernel Regression: Residuals, Fit & CV

- Now, the leave-one-out residuals are defined as:

$$e_{-i} = y_i - \hat{m}_{-i}(x_i), \quad i = 1, 2, ..., N.$$

e_{-i} is not a function of y_i ⇒ no tendency to overfit for small h.

- The mean squared leave-one-out residual is

$$CV(h) = \frac{1}{N} \sum_i^N (e_{-i}(h))^2$$

This function of h is known as the **cross-validation criterion**. This criterion can be used to select the bandwidth. It is common to trim the sample, to avoid the usual poor fitting at the tails.

- The CV bandwidth h_{CV} is the value that minimizes $CV(h)$. Usually, the restriction $h_{CV} \geq h_{LB}$ is imposed, where h_{LB} is a lower bound for h_{CV}, to make sure the bandwidth is not too small.

Kernel Regression: Residuals, Fit & CV

- The CV bandwidth h_{CV} is calculated numerically.

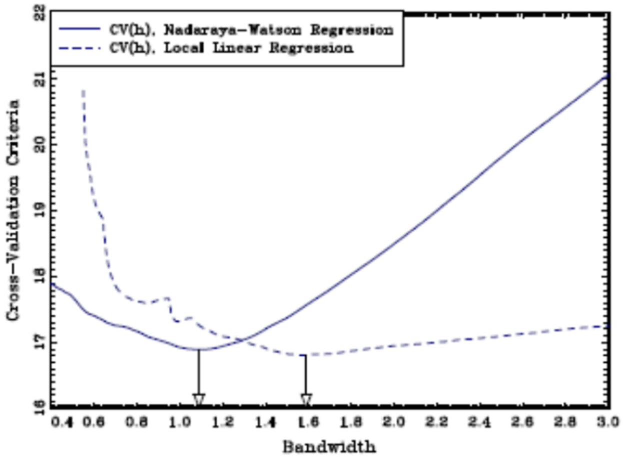
- A grid search is popular. Plots of $CV(h)$ against h are also used.

- It turns out that $CV(h)$ is an estimator of the mean-squared forecast error ($MSFE$). That is,

$$\text{E}[CV(h)] = MSFE_{N-1_1}(h) = MISE_{N-1}(h) + \sigma^2$$

Kernel Regression: Residuals, Fit & CV

- Plots of $CV(h)$against h for Hansen's simulated data for the NW and Local Linear estimators (with Epanechinikov kernel). From Hansen (2013).



Kernel Regression: NW Multivariate Estimator

- The **NW estimator** is defined by

$$\widehat{m}_h(x_0) = \frac{\sum_{i=1}^{N} K_h(x_i - x_0)\ y_i}{\sum_{i=1}^{N} K_h(x_i - x_0)} = \sum_{i=1}^{N} w_{N,h,i}(x_0)\ y_i$$

- The last expression simply shows that this estimator can be thought of as a weighted average of the observations of y. In matrix notation, we can write Ŷ = $\boldsymbol{M}(h)$ **Y**, with

$$M(h) = \begin{bmatrix} \frac{K\left(\frac{X_1 - x_1}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X_i - x_1}{h}\right)} & \frac{K\left(\frac{X_2 - x_1}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X_i - x_1}{h}\right)} & \cdots & \frac{K\left(\frac{X_n - x_1}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X_i - x_1}{h}\right)} \\ \frac{K\left(\frac{X_1 - x_2}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X_i - x_2}{h}\right)} & \frac{K\left(\frac{X_2 - x_2}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X_i - x_2}{h}\right)} & \cdots & \frac{K\left(\frac{X_n - x_2}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X_i - x_2}{h}\right)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{K\left(\frac{X_1 - x_n}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X_i - x_n}{h}\right)} & \frac{K\left(\frac{X_2 - x_n}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X_i - x_n}{h}\right)} & \cdots & \frac{K\left(\frac{X_n - x_n}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X_i - x_n}{h}\right)} \end{bmatrix}$$

Kernel Regression: NW Multivariate Estimator

- Kernel regression predictions: $\hat{Y} = \boldsymbol{M}(h)\, \boldsymbol{Y}$
- Liner regression predictions: $\hat{Y} = \mathbf{P}_x\, \boldsymbol{Y}$.

- A multivariate kernel is constructed, row by row, by computing the product of marginal densities for each variable in the matrix of regressors $\boldsymbol{X}$. That is,

$$h^{-d}K\left(\frac{X-x_i}{h}\right) = \prod_{j=1}^{d} h^{-1}K\left(\frac{x_j-x_{ji}}{h}\right)$$

- Usually, we use *leave-one-out* kernels. That is, the current observation is excluded in the kernel construction to avoid overfitting — the principal diagonal in $\boldsymbol{M}(h)$ is zeroes.

Comparison: Mean vs Kernel Smoother

- Mean (uniform) smoother

$$\widehat{m}_h(x_0) = \frac{\sum_{i=1}^{N} w(\frac{x_i - x_0}{h})\ y_i}{\sum_{i=1}^{N} w(\frac{x_i - x_0}{h})}$$

where

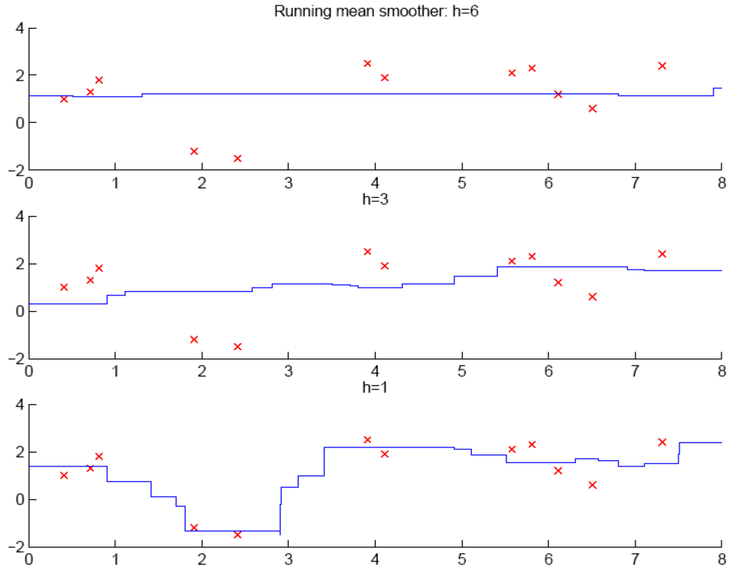
$$w(u) = \begin{cases} 1 & |u| < 1 \\ 0 & |u| \geq 1 \end{cases}$$

- Kernel smoother

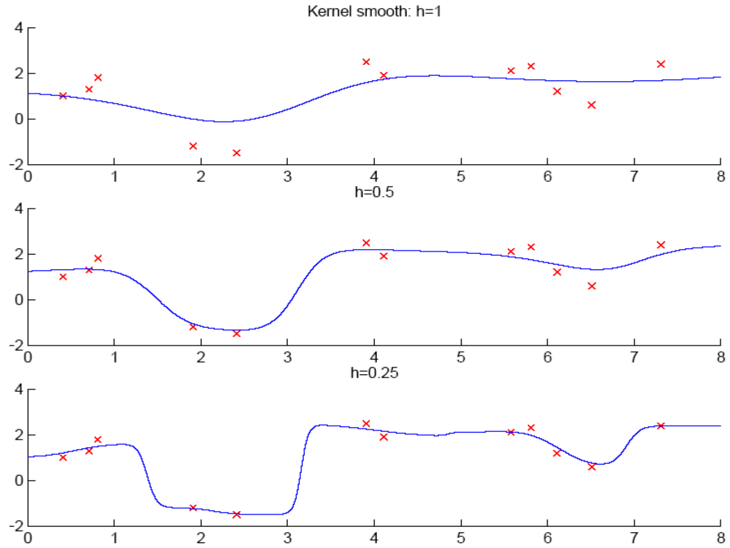
$$\widehat{m}_h(x_0) = \frac{\sum_{i=1}^{N} K(\frac{x_i - x_0}{h})\ y_i}{\sum_{i=1}^{N} K(\frac{x_i - x_0}{h})}$$

where $K(.)$ is Gaussian.

Comparison: Mean vs Kernel Smoother



Comparison: Mean vs Kernel Smoother



$\boldsymbol{k}$-Nearest Neighbor Estimator

- k-*NN* methods are more commonly used for regression than for density estimation. The classic *k-NN* smoother is defined as

$$\hat{m}_k(x_0) = \frac{1}{k}\sum_{i=1}^{N} I\big(\| x_i - x_0 \| \leq d_k(x_0)\big)\, y_i$$

This is the average value of y_i among the observations which are the *k* nearest neighbors of x_0. (d_k is the distance between x and x_0.)

- A smooth k-*NN* estimator is:

$$\hat{m}_k(x_0) = \frac{\sum_{i=1}^{N} w_k(\|x_i - x_0\| \leq d_k(x_0))\; y_i}{\sum_{i=1}^{N} w_k(\|x_i - x_0\| \leq d_k(x_0))} = \sum_{i=1}^{N} w_{N,k,i}(x_0)\;\; y_i$$

a weighted average of the k nearest neighbors.

$\boldsymbol{k}$-Nearest Neighbor Estimator

Example:

We have a sample $\{X, Y\}$={(1,5), (7,12), (3,1), (4,0), (5,4)}. Set k = 3. We want to calculate $\hat{m}_k(x_0 = 4)$ for the classic k-*NN* estimator, using Euclidian distance, $d_{k=3}(x_0 = 4) = 1$. Then, *Neighborhood*$(x_0 = 4) = \{3, 4, 5\}$.

The weights are $\{w_{N=5,k=3,i}(x_0 = 4)\}$ = {0, 0, 1/3, 1/3, 1/3}

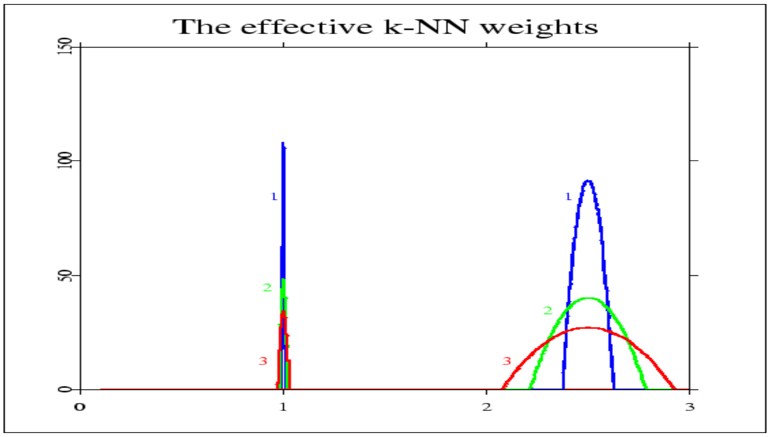
$$\hat{m}_k(4) = \frac{1}{k}\sum_{i=1}^{N} I(\| x_i - 4 \| \leq 1)\, y_i = \frac{(1+0+4)}{3} = 5/3$$

Note: If the X-variable is chosen from an equidistant grid, the k-*NN* weight are equivalent to kernel weights.

- If Epanechnikov weights are used, when observations get thin, the k-*NN* weights spread out more. See the food/income example, when $x_0 = 2.5$. (Very different weights from previous (fixed) case.)

k-Nearest Neighbor Estimator

Figure 4. The effective k-*NN* weights for the food versus net income data set. At $x_0 = 1$ and $x_0 = 2.5$ for k = 100 (label 1), k = 200 (label 2), k = 300 (label 3) with Epanechnikov kernel. From Hardle (1990).



k-Nearest Neighbor Estimator

- The smoothing parameter k regulates the degree of smoothness of the estimated curve. It plays a role similar to h for kernel smoothers.

- The influence of varying k on qualitative features of the estimated curve is similar to that observed for kernel estimation with a uniform kernel.

- When $k > N$, the k-*NN* smoother is equal to the average of the response variables. When k = 1, the observations are reproduced at x_i, and for an x between two adjacent predictor variables a step function is obtained with a jump in the middle between the two observations.

- When **X** is a vector, scaling matters. Then, always scale X.

k-Nearest Neighbor Estimator: Properties

- For the one regressor case, we have similar asymptotic results as in the univariate density case.

Let $N \to \infty$, $k \to 0$, and $Nk \to \infty$. Bias and variance of the k-NN estimate with *uniform* weights are given by

$$E[\hat{m}_k(x) - m(x)] \approx \frac{1}{24 f(x)^3}[(m'' f + 2m' f')(x)](k/n)^2$$

$$\text{var}\{\hat{m}_k(x)\} \approx \sigma^2(x)/k$$

Note: The optimal trade-off between bias^2 & Variance is achieved in an asymptotic sense by setting $k \sim N^{4/(4+q)}$, (q = dimension of $\boldsymbol{X}$).
⇒ when $q = 1$, $k \sim N^{4/5}$.

- If $k = 2Nh\, f(x)$we have exactly the same MSE at x for both kernel and k-*NN* estimators.

k-Nearest Neighbor Estimator: Properties

- For the multivariate case, the asymptotic analysis is the same as for density estimation.

- Conditional on $d_k(x_0)$; the bias and variance are approximately as for NW regression. The conditional bias is proportional to $d_k(x_0)$ and the variance to $1/[N\ d_k(x_0)^q]$ (q = dimension of $\boldsymbol{X}$).

- The optimal $k \sim N^{4/(4+q)}$ and the optimal convergence rate is the same as for NW estimation.

k-Nearest Neighbor Estimator: Computations

• A great advantage of the k-*NN* smoother is computational.

• Calculations can be easily updated. The algorithm requires O(N) operations to compute the smooth at all x_i*'s*. Compare this to O(N^2 h) calculations for the kernel estimator.

• Cross-validation is used to set k, using leave-one-out errors*:*

$$CV(k) = \frac{1}{N} \sum_i^N (e_{-i}(k))^2 = \frac{1}{N} \sum_i^N (y_i - \hat{m}_{-i}(x_i))^2$$

• Li (1987) and Andrews (1991) shows that it is asymptotically optimal to pick k by cross-validation.

Nonparametric Variance Estimation

Suppose we have the following DGP:

$y_i = \boldsymbol{x}_i{}'\,\boldsymbol{\beta} + m(\boldsymbol{x}_i, \boldsymbol{z}_i) + \varepsilon_i,$ $\quad\quad i = 1, 2, ..., N.$

$\text{E}[\varepsilon_i \,|\, \boldsymbol{x}_i, \boldsymbol{z}_i] = 0$

$\varepsilon_i{}^2 = \sigma^2(\boldsymbol{x}_i) + \eta_i,$ $\quad\quad \text{E}[\eta_i \,|\, \boldsymbol{x}_i] = 0$

- $\sigma^2(\boldsymbol{x}_i)$: the regression function of $\varepsilon_i{}^2$ on $\boldsymbol{x}_{i_i}$. We want to estimate it.

• Problem: If $\varepsilon_i{}^2$ were observed ⇒ NW or LL regression.

• Solution: Use the non- (or semi-) parametric residual, $e_i(k)$:

$$e_i(k) = y_i - \hat{m}_i(x_i))$$

• Then, we can use the NW estimator:

$$\hat{\sigma}(x_0) = \frac{\sum_{i=1}^N K(\frac{x_i - x_0}{h}) e_i{}^2}{\sum_{i=1}^N K(\frac{x_i - x_0}{h})}$$

Nonparametric Variance Estimation

- We have a **two-step estimator**. Similar situation if we use the LL estimator. The bandwidths h are not the same as for estimation of $\widehat{m}(x_0)$; although we use the same notation.

Note: the LL estimator is not guaranteed to be non-negative, while the NW (or weighted NW) estimator is always non-negative (if non-negative kernels are used).

- Fan and Yao (1998) derive the surprising result that the asymptotic distribution of this two-step estimator is identical to that of the one-step idealized estimator –i.e., using ε_i.

Series Estimation

- Series estimation is another nonparametric regression method. The idea is to approximate an unknown function, $m(x_0)$, with a flexible parametric function, with the number of parameters treated similarly to the bandwidth in kernel regression.

- A series approximation to $m(x_0)$ takes the general form:

$$m_k(x) = m_k(x, \beta)$$

where $m_k(x, \beta)$ is a known parametric family and β is a vector of k unknowns.

- A linear series approximation takes the form:

$$\widehat{m}_k = \sum_{j=1}^{K} z_{jk}(x)\beta_k = \boldsymbol{z}_j(x)'\beta_k$$

where $z_{jk}(x)$ are (nonlinear) functions of x, called **basis functions.**

Series Estimation

Example: The series approximation using a 3^{rd} order polynomial:

$$\hat{m}_k = \mathbf{z}_j(x)'\beta_k = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

In this case, the basis functions are $x,\ x^2$, and x^3.

Note: $\hat{m}_k = \mathbf{z}_j(x)'\beta_k$ is an example of a **linear basis expansion**.

Different basis functions produce different models:

- Linear model: $z_{jk}(x_j) = x_j$
- Polynomial model: $z_{jk}(x_j) = x^j$
- Piecewise constant: $z_{jk}(x_j) = I[t_1 \leq x_j \leq t_2]$
- Piecewise linear: $z_{jk}(x_j) = I[t_1 \leq \beta_0 + \beta_1 x_j \leq t_2]$

Series Estimation: Candidates

• Several candidates to use for series approximation:

(1) **Power series**. We can use a p^{th}order polynomial –i.e., $z_{jk}(x) = x^j$ It works well for low p's. But, they tend to be unstable for large p.

(2) **Trigonometric series** It produces bounded functions. It can produce wiggly, wild estimates.

(3) **Splines**. A continuous piecewise polynomial function. Splines can have any polynomial order (linear, quadratic, cubic, etc.). But, it is common to use a *cubic*. It is common to constrain the spline function to have continuous derivatives up to the order of the spline.

Series Estimation: Splines

Remark: So far, we have used (with kernel and k-NN regressions) local averages or local polynomials to do the estimation. Now, we move back to global models. With regression splines and smoothing splines, we build our estimator globally, from a set of **basis functions.**

• The basis functions are **splines**. There is more than one way to define a spline series expansion. All are based on the number of **knots** –the points between the segments.

• A (univariate) spline $m(x)$ of degree k with knots $t_1 < t_2 < \ldots < t_T$ is a piecewise polynomial of degree k that is continuous and has continuous derivatives of orders 1, …, $k-1$ at its knots. That is:

- $m(x)$ is a k^{th}degree polynomial on each $(\infty, t_1]$, $[t_1,\ t_2]$, …, $[t_T, \infty)$

- $D^l m(x)$ is continuous at each of t_1, …., t_T, for all l = 0, …, $k-1$.

Series Estimation: Splines

Example: A piecewise linear function, with 2 segments & a knot at t:

$$\widehat{m}_k = \begin{cases} m_1(x) = \beta_{00} + \beta_{01}(x-t) & x < t \\ m_2(x) = \beta_{10} + \beta_{11}(x-t) & x \geq t \end{cases}$$

The function $m_k(x)$ is continuous if $\beta_{00} = \beta_{10}$. Enforcing this (and transforming the coefficients), we get:

$$\widehat{m}_k = \beta_0 + \beta_1 x + \ \beta_2 (x-t)\ I[x \geq t]$$

Notes: Function has $k = 3$ coeficients –like a quadratic polynomial.

Also, the last term is an example of a **truncated power function**, which, in general, we write, with exponent j, as x_+^j:

$$x_+^j = \begin{cases} x^j & x > 0 \\ 0 & x \geq 0 \end{cases}$$

Series Estimation: Splines

• Following the above process, a piecewise quadratic function, with one knot and a continuous 1st derivative has $k = 4$. Similarly, a piecewise cubic function, with one knot and a continuous 2nd derivative has $k = 5$. The function $m_k(x)$ is

$$\widehat{m}_k(x) = \beta_0 + \beta_1\, x + \beta_2\; x^2 + \beta_3\; x^3 + \beta_4\; (x - t)_+^3$$

Note: The polynomial order p is selected to control the smoothness of the spline, as $m_k(x)$ has continuous derivatives up to $p - 1$.

• The approximation improves as the number of knots increases. For example, for a **cubic spline** with two knots t_1 & t_2 ($t_1 < t_2$). The form is:

$$\widehat{m}_k(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4\; (x - t_1)_+^3 + \beta_5 (x - t_2)_+^3$$

Series Estimation: Splines – Cubic

• Then, a p^{th} order spline with T knots at t_1, t_2, .., t_T (t_1<t_2<..< t_T) is

$$\widehat{m}_k = \sum_{j=0}^{p} \beta_j\; x^j + \sum_{t=1}^{T} \gamma_t\;\; (x - t_1)_+^p$$

which has $k = T + p + 1$ coefficients.

Remark (from Ryan Tibshirani): "It is said that a cubic spline is so smooth, that one cannot detect the locations of its knots by eye!"

• In spline approximations, the usual approach is to treat p as fixed, and select the number of knots, T, to determine the complexity of the approximation. Sometimes, the T knots are referred as "**interior knots**," to distinguish them from the endpoints of the sample.

• The t_t's are typically treated as fixed. It is common to set evenly spaced t_t's. When this happens, the knot sequence is called *uniform*.

Series Estimation: Splines – Regression

- For a given set of knots, the function $m_k(x)$ is linear in the parameters ⇒ LS estimation is possible!

Example: Suppose we use a cubic spline & two knots t_1 & t_2 ($t_1 < t_2$):

$$E[y|x, t_1, t_2) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4\,(x - t_1)_+^3 + \beta_5 (x - t_2)_+^3$$
$$= \boldsymbol{Z}\,\boldsymbol{\beta_k}$$

where $\boldsymbol{Z}$ is an Nxk matrix and is an kx1 with elements:

$$\boldsymbol{Z} = \begin{bmatrix} 1 & x_1 & {x_1}^2 & \dots & (x_1 - t_2)_+^3 \\ 1 & x_2 & {x_2}^2 & \dots & (x_2 - t_2)_+^3 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_N & {x_N}^2 & \dots & (x_N - t_2)_+^3 \end{bmatrix} \quad \& \quad \boldsymbol{\beta}_k = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_5 \end{bmatrix}$$

$$\Rightarrow \quad \widehat{\boldsymbol{\beta}}_k = (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\,\boldsymbol{Z}'\boldsymbol{y}$$

Then, the fitted values ("*linear smoother*"): $\widehat{\boldsymbol{y}} = \boldsymbol{Z}\,(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\,\boldsymbol{Z}'\,\boldsymbol{y}$

Series Estimation: B-Splines

- Another popular class of series approximation are called **B-splines**. ("***B***" for basis). They are basis functions which are bounded, integrable and density-shaped. They can be constructed or composed from a variety of basic shapes, usually polynomials.

- Let t be a sequence of knots –i.e., non-decreasing real number–, with $t_1 < t_2 < \dots < t_T$, which we augment with 2 m exterior knots: $t_{-(m-1)}$, …, t_0, t_{T+1}, ..., t_{T+m}.

- For each knot t_n, we recursively define the t-th B-spline basis function of degree j, $B_{t,j}$ (for j = 0, 1, … , r , where r is the order B-spline). For example, a 1st order B-spline is given by:

$$B_{t,1} = \begin{cases} 1 & \text{if } t_t \leq\ x < t_{t+1} \\ 0 & \quad\text{otherwise} \end{cases}$$

Series Estimation: B-Splines

- A 1st order B-spline is given by:

$$B_{t,1} = \begin{cases} 1 & \text{if } t_t \leq \ x < t_{t+1} \\ 0 & \quad \text{otherwise} \end{cases}$$

Note: A 1st order B-spline can form any piecewise constant function.

- We use a recursive formula to define a 2nd order B-spline:

$$B_{t,2}(x) = \alpha_{t,1}(x) * B_{t,1}(x) + (1 - \alpha_{t+1,1}(x)) * B_{t+1,1}\,(x)$$

where

$$\alpha_{t,1}(x) = \begin{cases} \frac{x - t_t}{t_{t+1} - t_t} & \text{if } t_t \neq \ t_{t+1} \\ 0 & \text{otherwise} \end{cases}$$

Series Estimation: B-Splines

- For the jth order B-spline we have:

$$B_{t,j+1}(x) = \alpha_{t,j+1}(x) * B_{t,j}(x) + (1 - \alpha_{t+1,j+1}(x)) * B_{t+1,j}\,(x)$$

where

$$\alpha_{t,j}(x) = \begin{cases} \frac{x - t_t}{t_{t+j} - t_t} & \text{if } t_t \neq \ t_{t+j} \\ 0 & \text{otherwise} \end{cases}$$ (if dividing by 0, set to 0)

- Convention: If dividing by 0, set basis element equal to 0.

- The first term, $B_{0,r}$ is sometimes called the "intercept." It can be set to zero to minimize multicollinearity problems.

Series Estimation: B-Splines Function

- The **B-spline function** of degree $(r-1)$ is a linear combination of the basis B-splines, $B_{t,r}$, of degree $(r-1)$:

$$B(x) = \sum_{t=1}^{T+r} \theta_n B_{t,r}(x|t_1, .., t_T) = \theta_{\boldsymbol{k}}'\boldsymbol{z}(x) \qquad x \in [t_0, t_{T+1}]$$

where $\boldsymbol{z}(x)$ is the vector of the basic functions.

- The θ_n (the "parameters") are usually called "control points." They can be estimated with LS.

- The number of basis functions, K, equals the sum of the degree of the B-spline basis functions & the number of interior knots plus one.

$\Rightarrow$ Dim(θ) = $K = T + r + 1$.

Series Estimation: B-Splines Function

- **Example**: A cubic-spline with T knots is given by:

$$\hat{m}_k = \sum_{t=1}^{T+4} \theta_t B_{t,4}(x|t_1, .., t_T) = \boldsymbol{z}(x)\ \boldsymbol{\theta_k}$$

which is a simple linear model with $K = T + 4$ parameters. It can be estimated with OLS:

$$\boldsymbol{z}(x) = \begin{bmatrix} B_{1,4}(x_1) & B_{2,4}(x_1) & \dots & B_{T+4,4}(x_1) \\ B_{1,4}(x_2) & B_{2,4}(x_2) & \dots & B_{T+4,4}(x_2) \\ \dots & \dots & \dots & \dots \\ B_{1,4}(x_N) & B_{2,4}(x_N) & \dots & B_{T+4,4}(x_N) \end{bmatrix} \ \&\ \boldsymbol{\theta}_k = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_{T+4} \end{bmatrix}$$

where $\boldsymbol{z}(x)$ is a Nx$(T+r)$ matrix. Then,

$$\hat{\theta}_k = (\boldsymbol{z}(x)'\boldsymbol{z}(x))^{-1}\ \boldsymbol{z}(x)'\boldsymbol{y}$$

- Now, the predictor ("*linear smoother*") is: $\hat{m}_j(x) = \boldsymbol{z}(x)\hat{\boldsymbol{\theta}}_k$

Series Estimation: B-Splines Function

• The j-th predictor is:

$$\hat{m}_j(x) = \boldsymbol{z}(x)\,\hat{\boldsymbol{\theta}}_k = \boldsymbol{z}(x)(\boldsymbol{z}(x)'\boldsymbol{z}(x))^{-1}\,\boldsymbol{z}(x)'\boldsymbol{y}$$

• We want $\hat{m}_j(x)$ to be flexible, we want to use lots of basis functions, say, setting $K = 15$ or larger. But, we penalize $\hat{m}_j(x)$ for being too "jerky."

• Common approach is to restrict (penalize) the second derivative – i.e., how much the slope of $\hat{m}_j(x)$ can change– over range of the predictor.

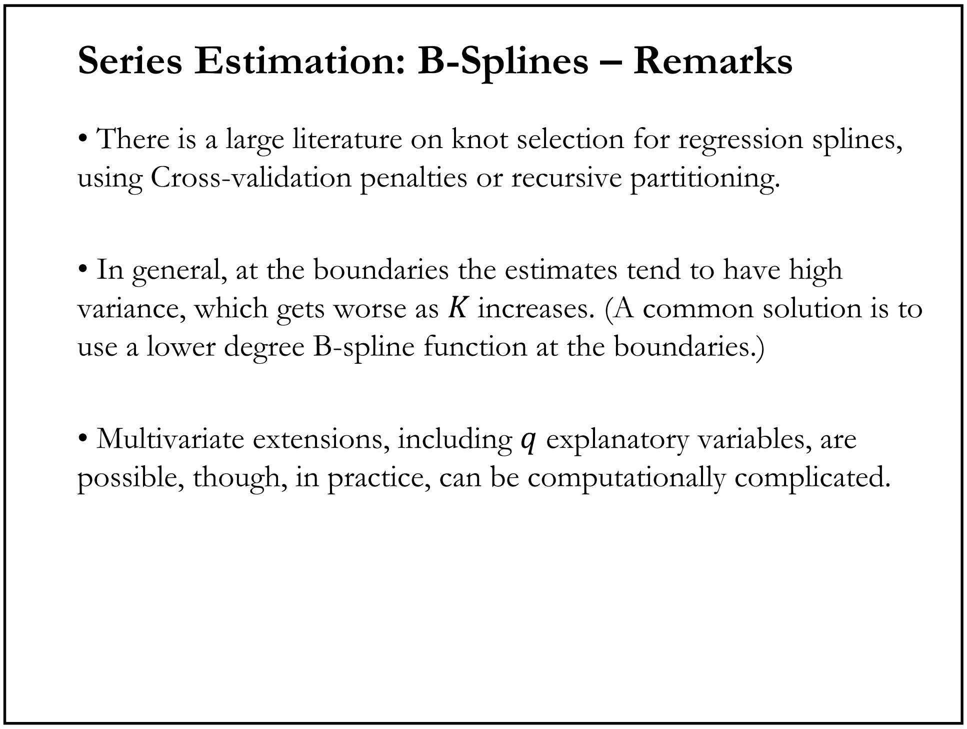
Series Estimation: B-Splines – Remarks

• A B-spline includes all polynomials of the same degree or less over $[t_0, t_{N+1}]$. We can think of B-splines regression as a generalization of polynomial regression.

• For example, a B-spline of order $r = 2$ includes all constant ($r = 0$), linear (r = 1), and quadratic (r = 2) functions over $[t_0, t_{T+1}]$ as special cases.

• It is not easy to choose the optimal number of knots and their locations, which is an infinite dimensional optimization problem.

• It is common to use equally spaced (uniform, equidistant) knots or set the knots equal to quartiles, for example, with $T = 3$, set $t_1 =$ 0.25th, $t_2 =$ 0.50th & $t_3 =$ 0.75th quartiles, respectively.

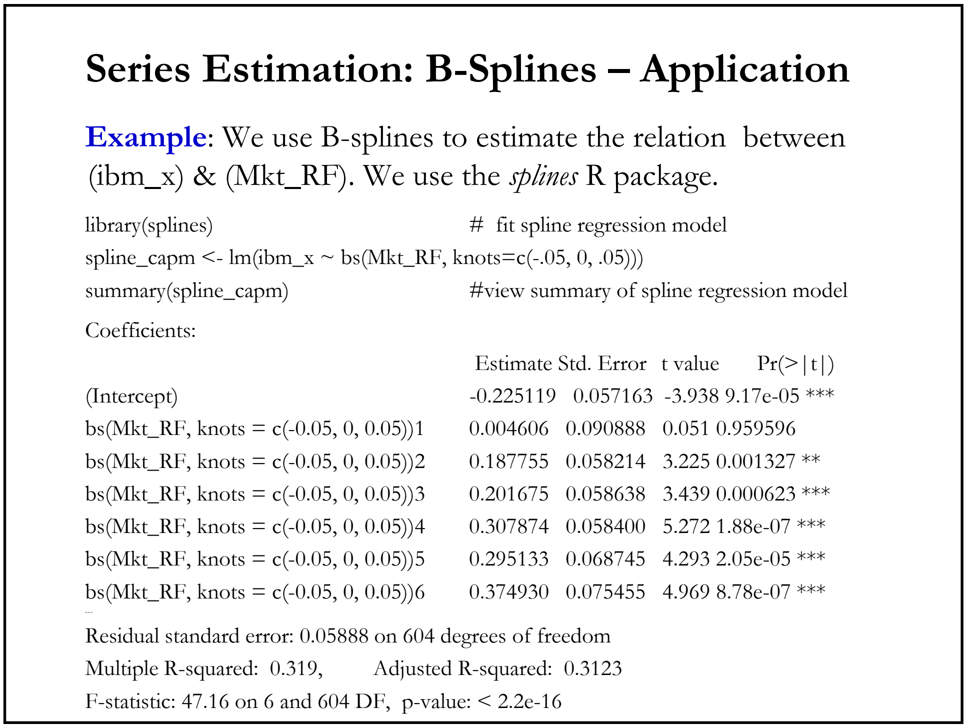


Series Estimation: B-Splines – Remarks

- There is a large literature on knot selection for regression splines, using Cross-validation penalties or recursive partitioning.

- In general, at the boundaries the estimates tend to have high variance, which gets worse as K increases. (A common solution is to use a lower degree B-spline function at the boundaries.)

- Multivariate extensions, including q explanatory variables, are possible, though, in practice, can be computationally complicated.



Series Estimation: B-Splines – Application

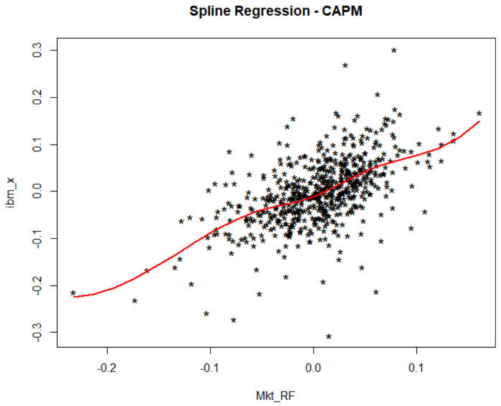
Example: We use B-splines to estimate the relation between (ibm_x) & (Mkt_RF). We use the *splines* R package.

```
library(splines)                                    #  fit spline regression model
spline_capm <- lm(ibm_x ~ bs(Mkt_RF, knots=c(-.05, 0, .05)))
summary(spline_capm)                                #view summary of spline regression model
Coefficients:
                                        Estimate Std. Error  t value     Pr(>|t|)
(Intercept)                            -0.225119   0.057163  -3.938 9.17e-05 ***
bs(Mkt_RF, knots = c(-0.05, 0, 0.05))1  0.004606   0.090888   0.051 0.959596
bs(Mkt_RF, knots = c(-0.05, 0, 0.05))2  0.187755   0.058214   3.225 0.001327 **
bs(Mkt_RF, knots = c(-0.05, 0, 0.05))3  0.201675   0.058638   3.439 0.000623 ***
bs(Mkt_RF, knots = c(-0.05, 0, 0.05))4  0.307874   0.058400   5.272 1.88e-07 ***
bs(Mkt_RF, knots = c(-0.05, 0, 0.05))5  0.295133   0.068745   4.293 2.05e-05 ***
bs(Mkt_RF, knots = c(-0.05, 0, 0.05))6  0.374930   0.075455   4.969 8.78e-07 ***
---
Residual standard error: 0.05888 on 604 degrees of freedom
Multiple R-squared:  0.319,        Adjusted R-squared:  0.3123
F-statistic: 47.16 on 6 and 604 DF,  p-value: < 2.2e-16
```

Series Estimation: B-Splines – Application

Example (continuation):

```
x_lim <- range(Mkt_RF)
i <- 2;
k <- 20;
x_r <- (abs(x_lim[1]) + x_lim[2])/k
x_v <- matrix(0,k+1,1)
x_v[1] <- x_lim[1]
while (i <= k+1 )
{
x_v[i] <- x_v[i-1] + x_r
i <- i + 1
}
```



```
newdata <- data.frame(Mkt_RF = x_v)
preds <- predict(spline_capm, newdata)
plot(Mkt_RF, ibm_x, cex=1.5, pch="*", main="Spline Regression - CAPM")
lines(x_v, preds, col="red", lwd=2)
```

Spline Smoothing

• Determination of K (or h) is not easy. A perfect fit can be achieved by giving a lot of local flexibility to $\hat{m}(x)$. The result of this flexibility will be a jerky, difficult to interpret $\hat{m}(x)$.

• **Spline smoothing** quantifies the competition between two goals:

- producing a good fit to the data –traditionally measured as SSR
- producing a good curve –i.e., without too much rapid local variation.

• The regression curve $\hat{m}_\lambda(x)$ is obtained by minimizing a penalized sum of squares:

$$\min \{S_\lambda(m) = \sum_{i=1}^{N}(y_i - m(x_i))^2 + \lambda \int (D^q m(x))^2 \, dx\}$$

for $q = K + 1$, where m is q-times differentiable function on $[a, b]$, and λ governs the trade-off between “fit” and jerkiness of the (q-1)th derivative of the curve m. The most common application is $K = 3$.

Spline Smoothing: Natural Spline

• When $K = 3$, the term $\int(m''(x))^2\, dx$ is a **roughness penalty**.

• The above minimization problem is an infinite-dimensional optimization problem over all functions $m(x)$ for which the criterion is well-defined and finite.

• It turns out that the problem over the class of all twice differentiable functions on $[a, b]$ has a unique solution: the **natural cubic spline**

• A **natural spline** $m(x)$ of degree k with knots $t_1 < t_2 < \ldots < t_T$ is a piecewise polynomial of degree k such that
- $m(x)$ is a k^{th}degree polynomial on each (∞, t_1], [t_1, t_2], …, [t_T, ∞)
- $m(x)$ is a $(k-1)^{th}$degree polynomial on each (∞, t_1], and [t_T, ∞)
- $D^l m(x)$ is continuous at each of t_1, …., t_T, for all l = 0, …, $k-1$.

Spline Smoothing: Natural Spline

• Implicitly, natural splines are only defined for an odd degree k (linear, cubic, etc.). At the boundary points, the behavior is different!

• The unique solution is the (natural) **cubic spline.** with knots at the input points $x_1, x_2, \ldots, x_N$

$\Rightarrow \widehat{m}_\lambda(x)$ is a cubic polynomial between two successive X-values.

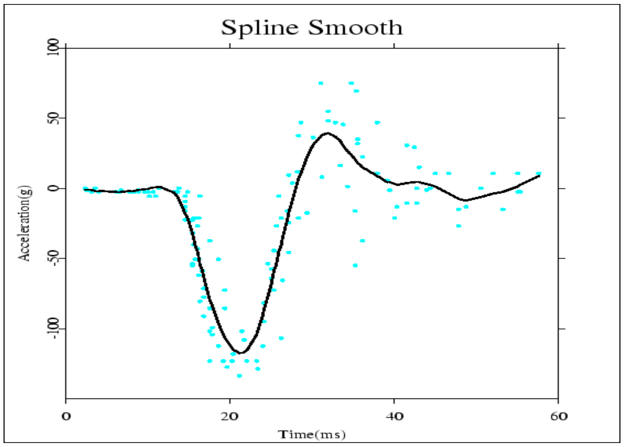
This result is known as the **Representer Theorem**; the proof is in Green and Silverman (1993).

• At the x_i, $\widehat{m}_\lambda(x)$ and its first two derivatives are continuous. At the boundary points x_1 and x_N, the second derivative is zero.

Note: These properties follow from the choice of penalty. A different penalty produces different solutions.

Spline Smoothing: Example

Figure 5. A spline smooth (Motorcycle data set). From Hardle (1990).



Spline Smoothing: Ridge Regression

- From the Representer Theorem, with $K = 3$, we pick a basis η_1, η_2, …, η_N for the set of 3rd degree natural splines with knots at x_1, x_2, …, x_N, and reparametrize the minimization problem (2) as

$$\min_\beta \sum_{i=1}^N (y_i - \sum_{j=1}^N \beta_j \eta_j(x_i))^2 + \lambda \int_a^b \{\sum_{j=1}^N \beta_j \eta_j''(x)\}^2 dx$$

- We solve for $\hat{\boldsymbol{\beta}} \in R^N$, the smoothing spline estimator is:

$$\widehat{m}_\lambda(x) = \sum_{i=1}^N \hat{\beta}_j(x)\eta_j(x_i)\, D^2$$

- Let basis matrix $\boldsymbol{L} \in R^{N\text{x}N}$ & penalty matrix $\boldsymbol{\Omega} \in R^{N\text{x}N}$ have entries

$$L_{ij} = \eta_j(x_i),\ \ \Omega_{ij} \approx \int_a^b D^2\eta_i(x) D^2\eta_j(x) dx$$

- Now, we can write the problem as:

$$\min_\beta \| \boldsymbol{y} - \boldsymbol{L\beta} \|_2^2 + \lambda\, \boldsymbol{\beta}' \boldsymbol{\Omega}\, \boldsymbol{\beta}$$

Spline Smoothing: Ridge Regression

- We write the problem as:

$$\min_{\beta} \parallel \boldsymbol{y} - \boldsymbol{L\beta} \parallel_2^2 + \lambda\, \boldsymbol{\beta}'\boldsymbol{\Omega}\, \boldsymbol{\beta}$$

which is similar to a ridge regression problem. Thus, the solution has the form:

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{L}'\boldsymbol{L} + \lambda\, \boldsymbol{\Omega})^{-1}\, \boldsymbol{L}'\boldsymbol{y}$$

- Then, we can write the smoothing spline as:

$$\begin{aligned}\widehat{m}_{\lambda}(x) &= \boldsymbol{L}(\boldsymbol{L}'\boldsymbol{L} + \lambda\, \boldsymbol{\Omega})^{-1}\boldsymbol{L}'\boldsymbol{y} \\ &= \textstyle\sum_{i=1}^{N} w_{\lambda,i}(x)\ y_i\end{aligned}$$

That is, the spline is linear in the y_i observations (a linear smoother).

- Q: Is the shape of the weights?

Spline Smoothing: Weights & Kernels

- Silverman (1984) showed for large N, small λ, and $x_i's$ not too close to the boundary,

$$w_{\lambda,i}(x, x_i) \approx \frac{1}{f(x_i)}\frac{1}{h(x_i)}K(\frac{x - x_i}{h(x_i)})$$

where the local bandwith $h(x_i)$ satisfies

$$h(x_i) = \frac{\lambda}{f(x_i)}^{-1/4}$$

and $K(.)$ is the "Silverman kernel":

$$K(z) = \frac{1}{2}\exp\left(-\frac{|z|}{\sqrt{2}}\right) * \sin(\frac{|z|}{\sqrt{2}} + \frac{\pi}{4})$$

- That is, the weight function looks like a kernel, an **equivalent kernel**.

Spline Smoothing: Weight Function – Example

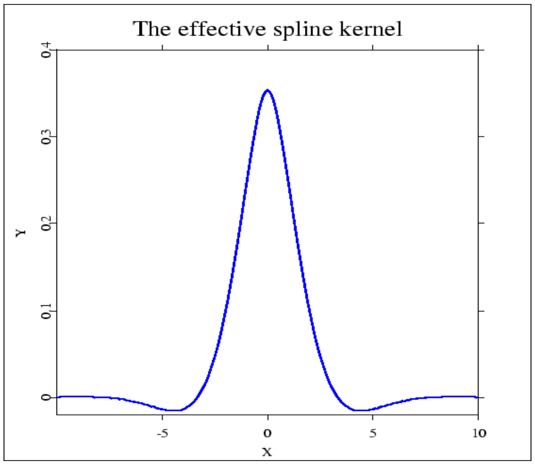


Figure 6. The asymptotic spline kernel function. From Hardle (1990).

$$K(z) = \frac{1}{2}\exp\left(-\frac{|z|}{\sqrt{2}}\right) * \sin(\frac{|z|}{\sqrt{2}} + \frac{\pi}{4})$$

Spline Smoothing: Equivalent Formulation

- A variation to compute splines is to solve the equivalent problem

$$min_m \int |m''(x)|^2 \, dx \qquad \text{subject to } \sum_{i=1}^{N}(y_i \; - \; m(x_i))^2 \leq \Delta$$

- The parameters λ and Δ have similar meanings, and are connected by the relationship

$$\lambda = -|G(\Delta)|^{-1}$$

where

$$G(\Delta) = \int (\widehat{m}_\Delta''(x))^2 \, dx$$

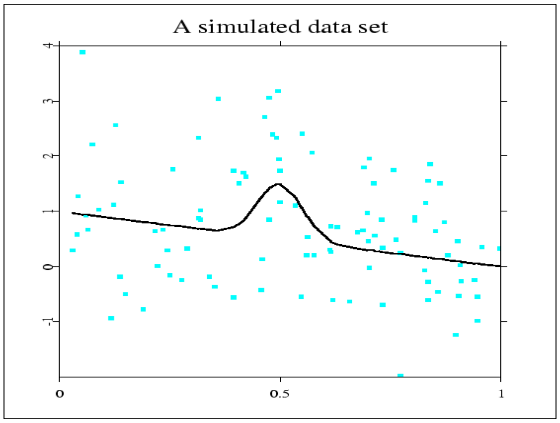
and $\widehat{m}_\Delta(x)$ solves the above problem.

Comparison: Kernel, *k*-NN & Spline Smoothers

Table 1. Bias and variance of kernel and k-*NN* smoother

	kernel	k-*NN*
bias	$h^2 \frac{(m''f+2m'f')(x)}{2f(x)} d_K$	$(k/n)^2 \frac{(m''f+2m'f')(x)}{8f^3(x)} d_K$
variance	$\frac{\sigma^2(x)}{nhf(x)} c_K$	$\frac{2\sigma^2(x)}{k} c_K$

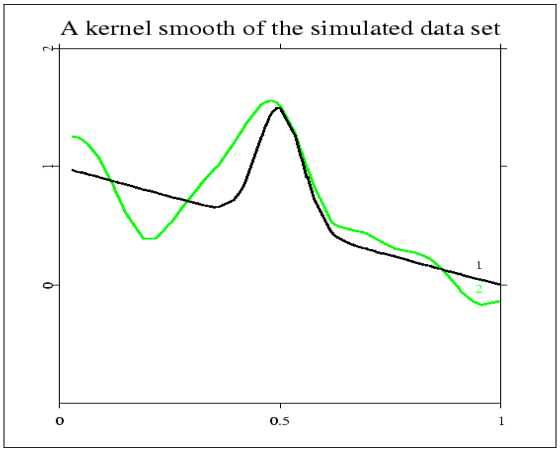
Comparison: Kernel, *k*-NN & Spline Smoothers



<u>Note</u>: Noisy Data.

Figure 7. Hardle (1990). A simulated data set. The raw data *N*=100 were constructed from $Y_i = m(X_i) + \varepsilon_i,\ \varepsilon_i \sim N(0,1), X_i \sim U(0,1)$ and $m(x) = 1 - x + e^{-200(x-1/2)^2}$

Comparison: Kernel, k-NN & Spline Smoothers

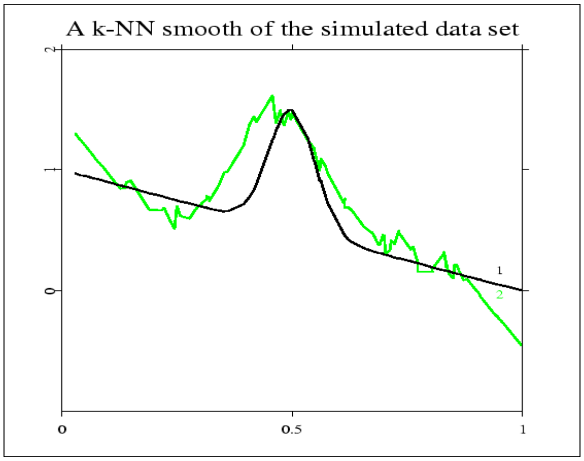


Note: As expected, kernel goes through the data.

Check smoother at boundaries (inaccurate at left).

Figure 8. A kernel smooth of the simulated data set. The black line (label 1) denotes the underlying regression curve $m(x) = 1 - x + e^{-200(x-1/2)^2}$ The green line (label 2) is the Gaussian kernel smooth $\hat{m}_h(x)$, h=.05

Comparison: Kernel, k-NN & Spline Smoothers

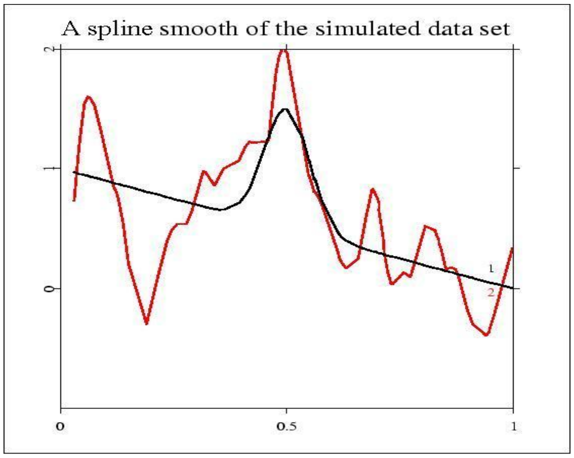


Note: Rougher curve..

Check smoother at boundaries (more points averaged).

Figure 9. Hardle (1990). A k-*NN* kernel smooth of the simulated data set. The black line (label 1) denotes the underlying regression curve. The green line (label 2) is the k-*NN* smoother. $\hat{m}_k(x)$, k=11.

Comparison: Kernel, k-NN & Spline Smoothers

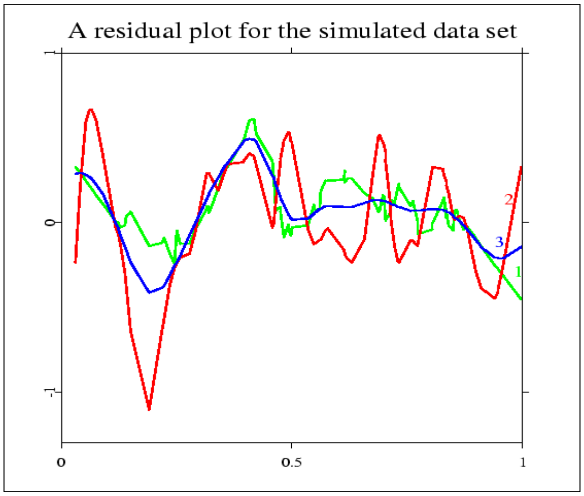


Note: As expected, very good track of observations.

Negative smooth (possible, even when all observations positive, check weights).

Figure 10. Hardle (1990). A spline smooth of the simulated data set. The black line (label 1) denotes the underlying regression curve. The green line (label 2) is the spline smoother $\hat{m}_{\Delta}(x)$, Δ=75.

Comparison: kernel, k-NN & Spline smoothers



Note: Similar overall pattern. Artificial bump at $x \approx 0.2$.

Figure 11. Hardle (1990). Residual plot of k-NN, kernel and spline smoother for the simulated data set.

Series Estimation: Polynomial Approximations

- A good series approximation $m_k(x)$ will have the property that it gets close to the true $m(x)$ as k increases.

- The **Stone-Weierstrass theorem**, (Weierstrass (1885), Stone (1937, 1948)) states that any continuous function can be arbitrarily uniformly well approximated by a polynomial of sufficiently high order:

$$\sup_{x \in X} |m_k(x) - m(x)| \leq \varepsilon$$

for any $\varepsilon > 0$.

- Thus, $m(x)$ can be arbitrarily well approximately by selecting a suitable polynomial. A usual choice is a cubic polynomial ($k = 3$):

$$m_k(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

Note: In this case, the basis functions are $x,\ x^2$, and x^3.

Series Estimation: Polynomial Approximations

- The above result can be strengthened. If the s-*th* derivative of $m(x)$ is continuous, then the uniform approximation error, $r_{k,i}$, satisfies

$$\sup_{x \in X} |r_{k,i} = m_k(x) - m(x)| = O(\, K^{-\alpha})$$

as $K \rightarrow \infty$ where $\alpha = s/d$. (dim$(X) = N \ * d$)

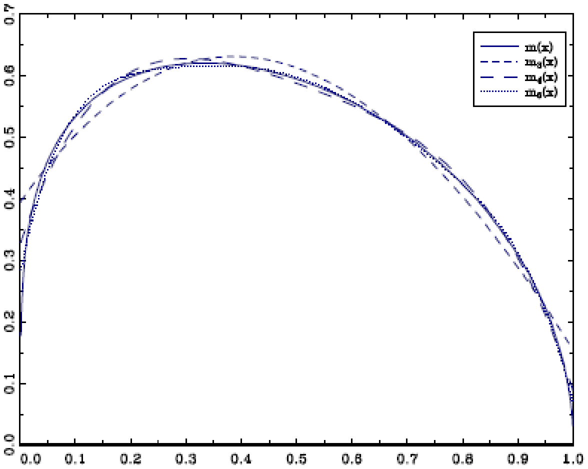
- Useful result: It gives a rate at which the approximation $m_k(x)$ approaches $m(x)$ as K increases.

- Intuitively, the number of derivatives s indexes the smoothness of $m(x)$. The best rate at which a polynomial (or spline) approximates $m(x)$ depends on the underlying smoothness of $m(x)$.

- Both results hold for spline approximations.

Series Estimation: Polynomial Approximations

- $m(x)$ can be arbitrarily well approximately by picking a suitable polynomial. We plot approximations of $m(x) = x^{.25}\,(1-x)^{.5}$ on $[0,1]$.



Note: The approximation with K = 3 is fairly crude, but improves with K = 4 and it is very good with K = 6.

Series Estimation: Polynomial Approximations

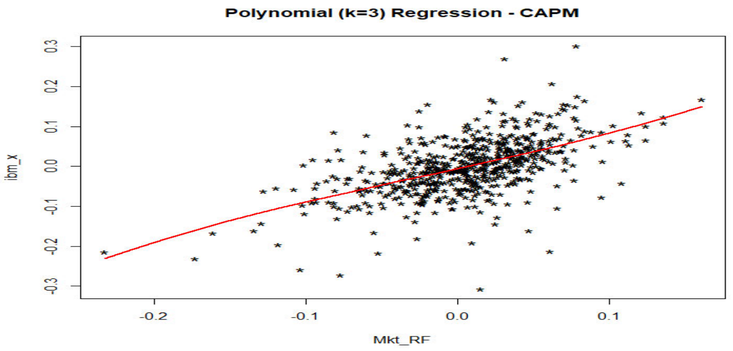
Example: We use 3rd degree polynomial to estimate the relation between (ibm_x) & (Mkt_RF). We use the *splines* R package.

```
x <- Mkt_RF
y <- ibm_x
mod_poly <- lm(y ~poly(Mkt_RF,3))
> summary(mod_poly)

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      -0.000219  0.002386  -0.092    0.927
poly(Mkt_RF, 3)1  0.980807  0.058965  16.634   <2e-16 ***
poly(Mkt_RF, 3)2  0.003651  0.058965   0.062    0.951
poly(Mkt_RF, 3)3  0.046379  0.058965   0.787    0.432
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05897 on 607 degrees of freedom
Multiple R-squared:  0.3136,      Adjusted R-squared:  0.3102
F-statistic: 92.43 on 3 and 607 DF,  p-value: < 2.2e-16
```

Series Estimation: Polynomial Approximations

Example (continuation):

```
newdata <- data.frame(Mkt_RF = x_v)
preds <- predict(mod_poly, newdata)
pred_capm <- predict(capm_ibm, newdata)
plot(Mkt_RF, ibm_x, cex=1.5, pch="*", main="Polynomial (k=3) Regression - CAPM")
lines(x_v, preds, col="red", lwd=2)
```



Series & Polynomials: Runge's Phenomenon

• Despite the excellent approximation implied by the Stone-Weierstrass theorem, polynomials have the troubling disadvantage that they are very poor at simple interpolation.

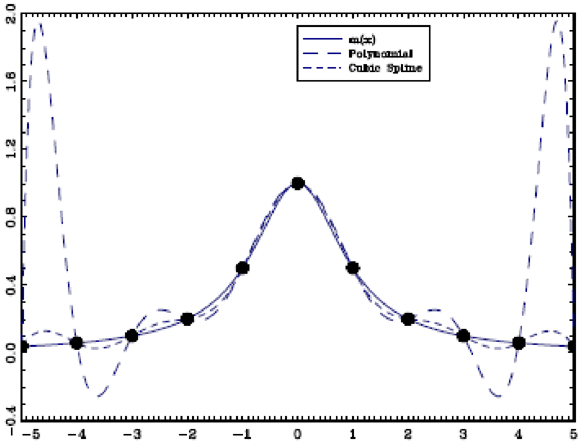
The problem is known as **Runge's phenomenon**.

• In contrast, splines do not show Runge's phenomenon. (See next Figure.) While the fitted spline displays some oscillation relative to $m(x)$, but they are relatively small.

• Because of Runge's phenomenon, high-order polynomials are not used for interpolation, and are not popular choices for high-order series approximations. Instead, splines are widely used.

Series & Polynomials: Runge's Phenomenon

- We plot approximations of $m(x_i) = 1/(1 + {x_i}^2)$ on [-5, 5], with K=11. Using a 10-*th* order polynomial. The discrepancy increases to infinity with K.



Note: The approximation is not accurate and far from the smoother true $m(x_i)$.

Series & Polynomials: Regression

- We have observations on (Y, X). Steps:

(1) For each i, and given K, construct the regressor vector $z_{k,i} = z_k(x_i)$, using the series transformations.

(2) Stack the observations in the matrices $\boldsymbol{y}$ and $\boldsymbol{z}_k$.

(3) Do OLS $\quad \Rightarrow b = (\boldsymbol{z}_k{}'\boldsymbol{z}_k)^{-1}\boldsymbol{z}_k{}'\,\boldsymbol{y}$

(4) Compute the LS regression function: $\hat{m}_k(x) = \boldsymbol{z}_k(x)'b_k$

(5) Compute estimated errors: $e_{k,i} = y_{k,i} - \hat{m}_k(x_i) = y_{k,i} - \boldsymbol{z}_k(x)'b_k$

Note: We estimate one error, $\varepsilon_{k,i}$, but we have two errors: the usual model error, ε_i, and the approximation error, $r_k(x_i) = r_{k,i}$. That is,

$$\varepsilon_{k,i} = r_{k,i} + \varepsilon_i$$

- To assess the fit of the regression, we can calculate the R^2 as usual.

Series & Polynomials: Regression – K

- β_k is a function of K. This reflect the goal to be flexible to incorporate greater complexity when the data are sufficiently informative. That is, K will typically be increasing with sample size N.

- K plays the role of h in kernel estimation. Larger K implies smaller approximation error but increased estimation variance.

- The number of series terms, K, can be determined through CV.

- The asymptotics are complicated. We need a new set of assumptions: Compact set, smoothness of $m(x)$, bounded error variance, non singularities in z_k, bounded E[$z_{k,i}{}' z_{k,i}$], K is chosen appropriately –i.e, a function of N and grows slower than N, etc.

Series & Polynomials: Regression – Asymptotics

- **Convergence**

Under certain assumptions (compact set, smoothness of $m(x)$, bounded error variance, non singularities in z_k, bounded E[$z_{k,i}{}' z_{k,i}$], K is chosen appropriately –i.e, a function of N and grows slower than N, etc.), the LS estimator $\boldsymbol{b}_k$ converges to β_k in *m.s.* distance. See Newey (1997).

- **Asymptotic normality**

Even though we are in a situation similar to parametric estimation, the fact that K can grow and the finite sample bias due to the approximation error, a new theory needs to be developed.

It turns out that under the same assumptions needed for convergence and imposing some mild restrictions on K and the bias, the estimator is asymptotically normal. See Newey (1997).

Series & Polynomials: Regression – Asymptotics

- The estimator has the asymptotic bias component $r_k(x)$, due to the finite order series as approximation to the unknown $m(x)$. The asymptotic distribution shows that the bias term is negligible if K diverges fast enough so that $NK^{-2\alpha} \to 0$. (In practical terms, this means that K is larger than optimal.)

Asymptotic standard errors for the $m(x)$ can be estimated with:

$$\hat{s}(x) = \sqrt{\frac{1}{n} z_K(x)' \widehat{Q}_K^{-1} \widehat{\Omega}_K \widehat{Q}_K^{-1} z_K(x)}.$$

where

$$\widehat{\Omega}_K = \frac{1}{n}\sum_{i=1}^{n} z_{Ki} z_{Ki}' \hat{e}_{iK}^2$$

$$\widehat{Q}_K = \frac{1}{n}\sum_{i=1}^{n} z_{Ki} z_{Ki}'.$$

See Newey (1997) for details.

Semiparametric Methods (SPM)

- A model is called ***semiparametric*** if it is described by θ and τ, where θ is finite-dimensional (parametric) and τ is infinite-dimensional (nonparametric).

- All moment condition models are semiparametric in the sense that the distribution of the data (τ) is unspecified and infinite dimensional. But the settings more typically called semiparametric are those where there is explicit estimation of τ.

- In many contexts the nonparametric part τ is a conditional mean, variance, density or distribution function.

- Often θ is the parameter of interest, and is a nuisance parameter, but this is not necessarily the case.

Semiparametric Methods (SPM): Example 1

Example: Feasible Nonparametric GLS

DGP: $\boldsymbol{y} = \boldsymbol{X}\,\theta + \varepsilon$ (dim($\boldsymbol{X}$)= Nxq)

E[$\boldsymbol{\varepsilon}$|$\boldsymbol{X}$] = 0

E[ε_i^2|$\boldsymbol{X}$] = $\sigma^2(x_i)$ ($\tau(x_i) = \sigma^2(x_i)$)

where the variance function $\sigma^2(x_i)$ is unknown but smooth in $\boldsymbol{X}$.

- We want to estimate $\boldsymbol{\theta}$. GLS is the efficient method, but it is not feasible.

Feasible GLS is possible. Replace $\sigma^2(x_i)$ using a nonparametric estimator (a kernel or a k-*NN* estimator).

- Q: What is the asymptotic distribution of the GLS estimator?

Semiparametric Methods (SPM): Example 2

Example: Generated Regressors

DGP: $y_i = \boldsymbol{\theta}\,\tau(x_i) + \varepsilon_i$

E[$\boldsymbol{\varepsilon}$|$\mathbf{X}$] = 0

$\boldsymbol{\theta}$ is finite dimensional and τ is an unknown function.

- Suppose τ is identified by another equation. We have consistent estimate, $\hat{\tau}(\boldsymbol{x})$. (Imagine a non-parametric Heckman estimator).

- Then, OLS is possible to estimate $\boldsymbol{\theta}$. This problem is called *generated regressors*, as the regressor is a (consistent) estimate of an infeasible regressor.

- Q: In general, the OLS estimator is consistent. But what is its distribution?

SPM: Asymptotic Distribution

- Based on Andrew's (1994) MINPIN paper.

Setting: $\hat{\boldsymbol{\theta}}$ MINimizes a criterion function, Q_N ($\boldsymbol{\theta}$, $\hat{\tau}$), which depends on a Preliminary Infinite dimensional Nuisance parameter estimator.

⇒ $\hat{\boldsymbol{\theta}}$ is a **two-step estimator**

- The usual derivation of asymptotic distributions expands the f.o.c. $m(\boldsymbol{\theta}, \tau) = \mathbf{0}$, We can do this for $\boldsymbol{\theta}$, but not for τ (it is infinite dimensional).

- To proceed, Andrews uses a stochastic equicontinuity assumption. Now, we work with the population version of $m(\theta, \tau)$ = E[$m_i(\theta, \tau)$] and study the convergence of

$$\upsilon_n(\tau) = \sqrt{N}\,(\bar{m}_n(\boldsymbol{\theta}_0, \tau) - m(\boldsymbol{\theta}_0, \tau))$$
$$= 1/\sqrt{N}\,\textstyle\sum_{i=1}^{N}\{m_i(\boldsymbol{\theta}_0, \tau) - \mathrm{E}[m_i(\boldsymbol{\theta}_0, \tau)]\}$$

SPM: Asymptotic Distribution

- Under a lot of assumptions: $\hat{\boldsymbol{\theta}}$ and $\hat{\tau}(\mathbf{x}) \xrightarrow{p}$ to $\boldsymbol{\theta}_0$ and τ_0; f.o.c. equal to 0 at $(\boldsymbol{\theta}_0, \tau_0)$ –i.e., identification condition-, convergence of f.o.c.; smoothness of underlying functions; and existence of moments),

$$\sqrt{N}\,(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0) \xrightarrow{d} N(0, \boldsymbol{V}),$$

where

$$\begin{aligned} V &= M^{-1}\Omega M^{-1\prime} \\ M &= \mathrm{E}\frac{\partial}{\partial\theta'}m_i(\theta_0, \tau_0) \\ \Omega &= \mathrm{E}m_i(\theta_0, \tau_0)\, m_i(\theta_0, \tau_0)' \end{aligned}$$

- The theorem says that $\hat{\boldsymbol{\theta}}$ has the same asymptotic distribution as the idealized estimator obtained by replacing the nonparametric estimate $\hat{\tau}$ with the true function τ_0.

⇒ the estimator is adaptive.

SPM: Asymptotic Distribution

• But the assumptions are not trivial. The convergence in probability assumptions need to be verified. The key assumption is

$$m(\boldsymbol{\theta}_0, \tau_0) = \delta Q_N(\boldsymbol{\theta}, \tau)/\delta\theta \,|(\boldsymbol{\theta} = \boldsymbol{\theta}_0, \tau = \tau_0) = \mathbf{0}.$$

• This assumption does not always hold. It turns out, it requires a sort of orthogonality condition between the estimation of $\boldsymbol{\theta}$ and τ.

• It holds for example 1 (FGLS with nonparametric variance), but not for Example 2 (generated regressors).

SPM: Partially Linear Regression Model

• It is easy to define a "**partially linear**" regression model:

$y_i \;=\; m_z(\boldsymbol{z}_i) + \boldsymbol{x}_i{}'\,\boldsymbol{\beta} + \varepsilon_i$ (dim($\boldsymbol{z}_i$)=Nxq)

$\text{E}[\varepsilon_i\,|\,\boldsymbol{x}_i, \boldsymbol{z}_i] = 0$

$\text{E}[\varepsilon_i{}^2|\,\boldsymbol{x}_i = \boldsymbol{x}, \boldsymbol{z}_i = \boldsymbol{z}] = \sigma^2(\boldsymbol{x}, \boldsymbol{z})$

- The regressors are $(\boldsymbol{x}, \boldsymbol{z})$.
- The conditional mean is linear in $\boldsymbol{x}_i$, but possibly non-linear in $\boldsymbol{z}_i$.
- Dummy variables are usually put in the $\boldsymbol{x}$ vector
- To keep things simple, we assume just one nonlinear variable: q =1.

• Goal: Estimate $\boldsymbol{\beta}$ and $m_z(\boldsymbol{z}_i)$, and to obtain C.I.

• Issues: Identification, Distribution of estimates.

SPM: Estimation

• Robinson (*Econometrica*, 1988) shows we can concentrate out $m_z(\boldsymbol{z}_i)$ by using a genearlization of residual regression. Start with:

$$y_i \;=\; m_z(\boldsymbol{z}_i) + \boldsymbol{x}_i{}'\,\beta + \varepsilon_i, \qquad (\text{dim}(\boldsymbol{z}) = N * q\,)$$

Taking conditional expectations n **Z**:

$$\text{E}[y_i|\boldsymbol{z}_i] = \text{E}[m_z(\boldsymbol{z}_i)|\boldsymbol{z}_i] + \text{E}[\;\text{x}_i{}'\,\beta\,|\,\boldsymbol{z}_i] = m_z(\boldsymbol{z}_i) + \text{E}[\boldsymbol{x}_i'|\boldsymbol{z}_i]\;\beta$$

- Two conditional means:

  - $m_y(\boldsymbol{z}_i) = \text{E}[y_i|\boldsymbol{z}_i]$
  - $m_x(\boldsymbol{z}_i) = \text{E}[\boldsymbol{x}_i|\boldsymbol{z}_i]$

- Then,

$$m_y(\boldsymbol{z}_i) = m_z(\boldsymbol{z}_i) + m_x(\boldsymbol{z}_i)'\,\beta$$

Subtract from the original equation ($m_z(\boldsymbol{z}_i)$ disappears):

$$y_i - m_y(\boldsymbol{z}_i) = [\boldsymbol{x}_i{}' - m_x(\boldsymbol{z}_i)']\;\beta + \varepsilon_i$$

SPM: Estimation

• Rewrite relation in terms of residuals:

$$y_i - m_y(\boldsymbol{z}_i) = [\boldsymbol{x}_i{}' - m_x(\boldsymbol{z}_i)']\;\beta + \varepsilon_i$$

- $\varepsilon_{yi} = y_i - m_y(\boldsymbol{z}_i)$
- $\varepsilon_{xi} = [\boldsymbol{x}_i{}' - m_x(\boldsymbol{z}_i)']$
- $\varepsilon_{yi} = \varepsilon_{xi}{}'\,\beta + \varepsilon_i$

That is, β is the coefficient of the regression of ε_{yi} on ε_{xi}. But, we do not observe the errors. It is an unfeasible LS estimator!

• Robinson suggests the following steps:

1) Estimate $m_y(.)$, $m_x(.)$ by NW/LL regression (different h's, OK).

2) Get the residuals, ε_{xi} & ε_{yi}.

3) Using the residuals, do OLS to estimate β.

SPM – Estimation: Trimming

• The nonparametric regression estimates depend inversely on $\widehat{f}_z(z)$.

Problem: For values of $\boldsymbol{z}_i$ where $\widehat{f}_z(\boldsymbol{z}_i)$ is close to 0. $\widehat{f}_z(z)$ is not bounded away from 0. The NW estimates at this point can be poor.

Solution: Trimming.

Let $\boldsymbol{b} > 0$ be a trimming constant. The trimmed estimator of β is:

$$\hat{\beta} = (\sum_{i=1}^{N} \varepsilon_{xi}\varepsilon_{xi}' \, \mathrm{I}[\widehat{f}_z(\boldsymbol{z}_i) \geq 0])^{-1} \sum_{i=1}^{N} \varepsilon_{xi}\varepsilon_{yi}' \, \mathrm{I}[\mathrm{f}_z\widehat{f}_z(\boldsymbol{z}_i) \geq 0]$$

⇒ This is a trimmed LS residual regression.

The asymptotic theory requires that $\boldsymbol{b} = \boldsymbol{b}_N \to 0$, but it is not clear how to select $\boldsymbol{b}$ in practice. Often trimming is ignored in applications.
Suggestion: Estimate model with and without trimming.

SPM – Asymptotic Distribution

• The needed regularity conditions: the data are *i.i.d.*, $\boldsymbol{z}_i$ has a density, and the regression functions, density, and conditional variance function are sufficiently smooth with respect to their arguments.

• Assume h is the same for all q. The important condition on the h sequence is

$$\sqrt{n}\left(h^4 + \frac{1}{nh^q}\right) \to 0$$

• Equivalently, what is essential is that the estimators themselves converge faster than $N^{-1/4}$. From the theory for nonparametric regression, these rates hold when h*'s* are picked optimally and $q \leq 3$.

SPM – Asymptotic Distribution

Theorem (Robinson). Under regularity conditions, including $q \leq 3$; the trimmed estimator satisfies

$$\sqrt{n}\left(\hat{\beta}-\beta\right) \rightarrow_d N(0, V)$$

$$V = \left(E\left(\epsilon_{xi}\epsilon_{xi}'\right)\right)^{-1}\left(E\left(\epsilon_{xi}\epsilon_{xi}'\sigma^2(X_i, Z_i)\right)\right)^{-1}\left(E\left(\epsilon_{xi}\epsilon_{xi}'\right)\right)^{-1}$$

That is, $\hat{\boldsymbol{\beta}}$ is asymptotically equivalent to the infeasible LS estimator.

- Estimate the variance matrix **V** as usual, using residuals.

SPM – Estimation of Nonparametric Part

- The model:

$$y_i \;=\; m_z(\boldsymbol{z}_i) + \boldsymbol{x}_i{}'\,\boldsymbol{\beta} + \varepsilon_i, \qquad (\dim(\boldsymbol{z}) = N * q\,)$$

- We estimated $\boldsymbol{\beta}$. Now, we want to estimate $m_z(\boldsymbol{z}_i)$. It looks like an iterative algorithm is needed, but since $\boldsymbol{\beta}$ converges faster than the nonparametric rate, we can pretend it is fixed. Then,

$$\hat{m}_z(z_0) = \frac{\sum_{i=1}^{N} K_h(\frac{z_0 - z_i}{h})(y_i - X_i{}'\hat{\beta})}{\sum_{i=1}^{N} K_h(\frac{z_0 - z_i}{h})}$$

- The bandwidth h = (h_1, ..., h_q) is distinct from those for the first-stage regressions. Standard errors for $m_z(\boldsymbol{z}_i)$ as usual for standard nonparametric regression.

SPM – Bandwidth Choice

• In a semiparametric context, it is important to study the effect a bandwidth has on the performance of the estimator of interest before determining the bandwidth.

• In many cases, this requires a nonconventional bandwidth rate.

• However, this problem does not occur in partially linear models. The first-step bandwidths h used for $\hat{m}_y(\boldsymbol{z}_i)$ and $\hat{m}_x(\boldsymbol{z}_i)$ are inputs for calculation of $\hat{\boldsymbol{\beta}}$.

• h impacts the theory for $\hat{\boldsymbol{\beta}}$, through the uniform convergence rates for $\hat{m}_y(\boldsymbol{z}_i)$ and $\hat{m}_x(\boldsymbol{z}_i)$, suggesting that we use conventional bandwidth rules, for example CV.

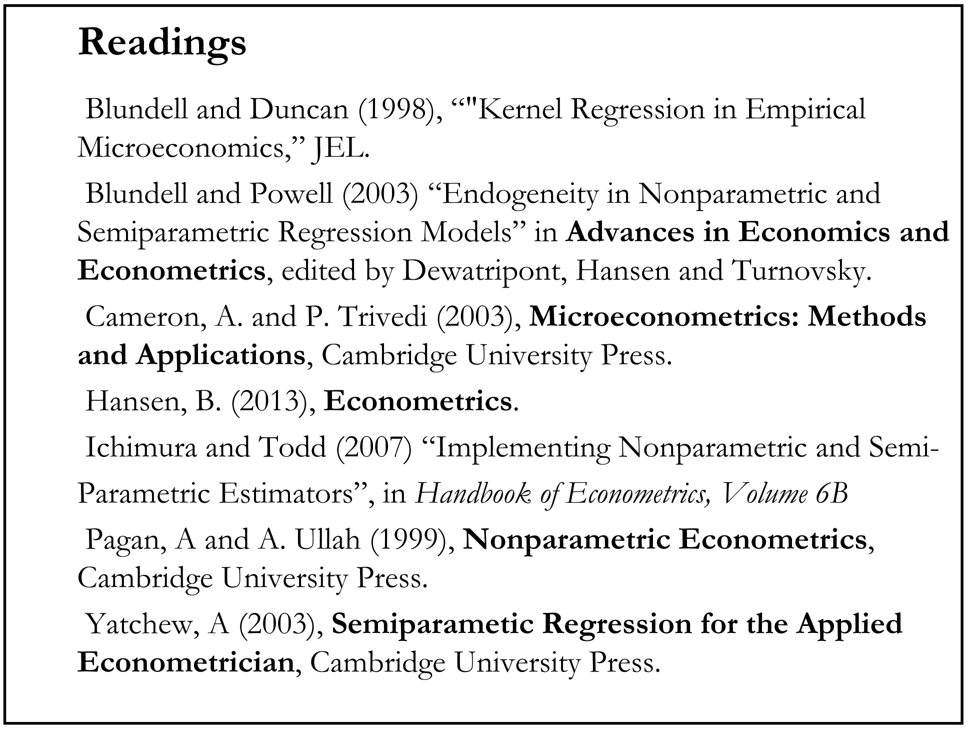
Further Comments

There are some specification tests that compare non-parametric regressions ("unconstrained" model) with parametric regressions ("constrained" model). See Blundell and Duncan (1998), Pagan and Ullah (1999) and Yatchew (Chapter 6).

• Recent research has focused on correcting for *endogeneity* (see Yatchew) and *heteroscedasticity* (see Yatchew). In general, the most promising approaches are two-step methods.

(1) Non-parametrically regress endogenous x variables on the IV $\boldsymbol{z}$, and calculate "errors" as the difference between those x variables and their (non-parametrically) predicted values.

(2) Add these errors into the equation of interest.



Readings

Blundell and Duncan (1998), “"Kernel Regression in Empirical Microeconomics,” JEL.

Blundell and Powell (2003) “Endogeneity in Nonparametric and Semiparametric Regression Models” in **Advances in Economics and Econometrics**, edited by Dewatripont, Hansen and Turnovsky.

Cameron, A. and P. Trivedi (2003), **Microeconometrics: Methods and Applications**, Cambridge University Press.

Hansen, B. (2013), **Econometrics**.

Ichimura and Todd (2007) “Implementing Nonparametric and Semi-Parametric Estimators”, in *Handbook of Econometrics, Volume 6B*

Pagan, A and A. Ullah (1999), **Nonparametric Econometrics**, Cambridge University Press.

Yatchew, A (2003), **Semiparametic Regression for the Applied Econometrician**, Cambridge University Press.