Lecture 15: Nonparemetric regression I

Pratheepa Jeganathan

05/06/2019

Recall

- One sample sign test, Wilcoxon signed rank test, large-sample approximation, median, Hodges-Lehman estimator, distribution-free confidence interval.
- Jackknife for bias and standard error of an estimator.
- Bootstrap samples, bootstrap replicates.
- Bootstrap standard error of an estimator.
- Bootstrap percentile confidence interval.
- Hypothesis testing with the bootstrap (one-sample problem.)
- Assessing the error in bootstrap estimates.
- Example: inference on ratio of heart attack rates in the aspirin-intake group to the placebo group.
- The exhaustive bootstrap distribution.

- Discrete data problems (one-sample, two-sample proportion tests, test of homogeneity, test of independence).
- Two-sample problems (location problem - equal variance, unequal variance, exact test or Monte Carlo, large-sample approximation, H-L estimator, dispersion problem, general distribution).
- Permutation tests (permutation test for continuous data, different test statistic, accuracy of permutation tests).
- Permutation tests (discrete data problems, exchangeability.)
- Rank-based correlation analysis (Kendall and Spearman correlation coefficients.)
- Rank-based regression (straight line, multiple linear regression, statistical inference about the unknown parameters, nonparametric procedures - does not depend on the distribution of error term.)
- Smoothing (density estimation, bias-variance trade-off, curse of dimensionality)

Nonparametric regression

Introduction

- Smoothers use external functions to model the functional relationship between y and x.
- External functions: lines or low order polynomial functions.
- Nonparametric
  - lack of a specific, parametric form assumed for the regression function being estimated.
  - no strong distributional assumptions on the errors.
- We will discuss the linear smoothers.
  - estimates are linear combinations of observed data.
- Local averaging, local regression, local polynomial, kernel smoothing, penalized regression.

Nonparametric regression

- We are given n pairs of observations $(x_1, Y_1), \cdots, (x_n, Y_n)$.
- Regression model
  - $Y_i = r(x_i) + \epsilon_i, i = 1, \cdots, n$.
    - Y response variable.
    - x covariate/feature.
    - $\mathbb{E}(\epsilon_i) = 0$.
    - r is a regression function.
- Estimation
  - Assume covariate value x_i are fixed.
    - If we treat x_i as random :
    - Data: $(X_1, Y_1), \cdots, (X_n, Y_n)$.
    - $r(x_i) = \mathbb{E}(Y|X = x)$, mean of Y conditional on $X = x$.
  - Assume $\mathbb{V}(\epsilon_i) = \sigma^2$ does not depend on x.
  - Estimate of $r(x)$ is $\hat{r}_n(x)$, smoother.

Linear smoother

Linear smoother

- Linear smoothers: estimates are linear combinations of observed data.
- An estimator $\hat{r}_n$ of r is a linear smoother if, for each x, $\exists$ a vector $\boldsymbol{l}(x) = (l_1(x), \cdots, l_n(x))^T$ such that

$$\hat{r}_n(x) = \sum_{i=1}^{n} l_i(x) Y_i.$$

- Define the vector of fitted values

$$\boldsymbol{r} = (\hat{r}_n(x_1), \cdots, \hat{r}_n(x_n))^T.$$

- It follows that

$$\boldsymbol{r} = \mathbf{L}\boldsymbol{Y},$$

where $\boldsymbol{Y} = (Y_1, \cdots, Y_n)^T$ and $L_{ij} = l_j(x_i), i = j = 1, \cdots, n$ and $\mathbf{L}$ is an $n \times n$ matrix.

Linear smoother

- The i-th row in $\mathbf{L}$ is the weights given to each Y_i in forming the estimate $\hat{r}_n(x_i)$.
- L is called the smoothing matrix or the hat matrix.
- The i-th row of $\mathbf{L}$ - effective kernel for estimating $r(x_i)$.
- $\nu = \mathrm{tr}(\mathbf{L})$ - effective degrees of freedom.
- For all x, $\sum_{i=1}^{n} l_i(x) = 1$ (i.e., if $Y_i = c \ \ \forall i$, then $\hat{r}_n(x) = c$.)

Some linear smoothers

Regressogram

- From **W2006**.
- Mostly like histogram.
- Suppose $a \leq x \leq b, \quad i = 1, \cdots, n$.
- Dived (a, b) in to $B_1, \cdots, B_m$ equally spaced bins.
- Let k_j be number of points in B_j. $\hat{r}_n$ is obtained by averaging Y_i's over each bin.

$$\hat{r}_n(x) = \frac{1}{k_j} \sum_{i:x_i \in B_j} Y_i \quad for \quad x \in B_j.$$

- We can write $\hat{r}_n(x) = \sum_{i=1}^{n} l_i(x) Y_i$.

Regressogram

- From **W2006** Page 67, 5.24 Example.
- Example: Let $n = 9, m = 3$ and $k_1 = k_2 = k_3 = 3$. Then,

$$\mathbf{L} = \frac{1}{3} \times \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

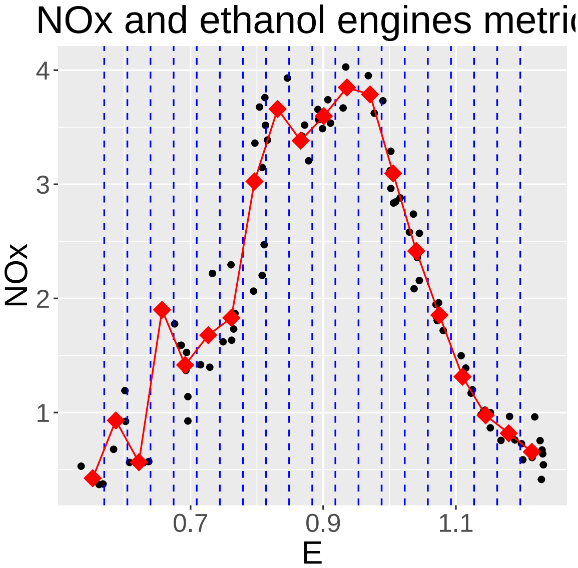
- Effective number of freedom $\nu = \mathrm{tr}\left(\mathbf{L}\right)$.
- Binwidth $h = \dfrac{b-a}{m}$ controls the smoothness of the estimate.

Regressogram

- **HWC** (Page 662) Example 14.2: Nitrogen Oxide Concentrations
  - Brinkman (1981) collected data on the nitrogen oxide concentrations ($\boldsymbol{Y}$) found in engine exhaust for ethanol engines with various equivalence ratios ($\boldsymbol{x}$).

```
library(NSM3)
data("ethanol")
library(HoRM)
library(ggplot2)
p = regressogram(ethanol$E, ethanol$NOx,
  nbins = 20, show.bins = TRUE,
              show.means = TRUE, show.lines = TRUE,
  x.lab = "E", y.lab = "NOx",
  main = "NOx and ethanol engines metric") +
  theme(plot.title = element_text(hjust = 0.5))
```

Regressogram



- blue dots are bins.

Local averaging (Friedman)

- **HWC** Chapter 14.1
- Estimate of r at the point x_i is taken to be the average of observed values Y_j corresponding to values x_j in some vicinity of x_i.
- The neighborhood of x_i is chosen to be the smallest symmetric window about x_i containing fixed number of observations.
- The average is a linear combination of the points in the neighborhood, thus, the fit is a linear smoother.
- `supsmu` function in R.

Local averaging

- From **W2006** Page 68, 5.26 Example.
- For $h > 0$ and let $B_x = \{i : |x_i - x| \leq h\}$.
- Let n_x be the number of points in B_x.
- $\hat{r}_n(x) = \frac{1}{n_x} \sum_{i \in B_x} Y_i$.
- We can write

$$\hat{r}_n(x) = \sum_{i=1}^{n} l_i(x) Y_i, \tag{1}$$

where $l_i(x) = \frac{1}{n_x}$ if$|x_i - x| \leq h$ and 0 otherwise.

- Example: Suppose $n = 9$, $x_i = \frac{i}{9}$ and $h = \frac{1}{9}$.

Local averaging

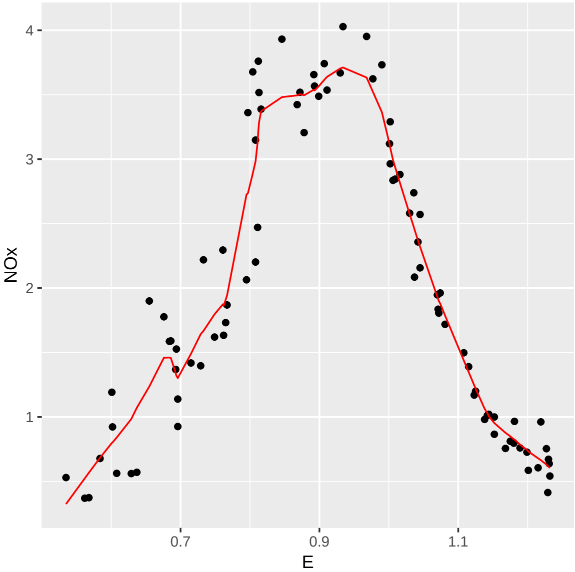
-

$$\mathbf{L} = \begin{pmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 & & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \end{pmatrix}$$

Local averaging

```
fit.local.avg = supsmu(ethanol$E,
  ethanol$NOx, span = "cv")
df.fit.local.avg = data.frame(E.fit = fit.local.avg$x,
  NOx.fit = fit.local.avg$y)
library(dplyr)
p = ggplot() +
  geom_point(data = ethanol, aes(x = E, y=NOx)) +
  geom_line(data = df.fit.local.avg,
    aes(x = E.fit, y = NOx.fit), color = "red")
```

Local averaging



Local regression (Cleveland)

- **HWC** Chapter 14.2
- Estimate r by performing a local linear regression (locally weighted least squares) on the observations (x, Y) near x_i.
- The regression is a weighted regression - weights are related to the distance of the points used in the regression to the point x_i.
- `loess` function in R, `loess.as` for cross-validation and finding optimal span.
  - The weight function used in `loess` is tricube function:

$$W(x) = \left(1 - |x|^3\right)^3, |x| < 1.$$

  - Let $w_1^i, \cdots, w_n^i$ be the weights determined by the centered and scaled W for a particular point x_i.
  - The weighted local regression is found by minimizing

$$\sum_{j=1}^{n} w_j^i \left(Y_i - \beta_0^i - \beta_1^i x_j\right)^2.$$

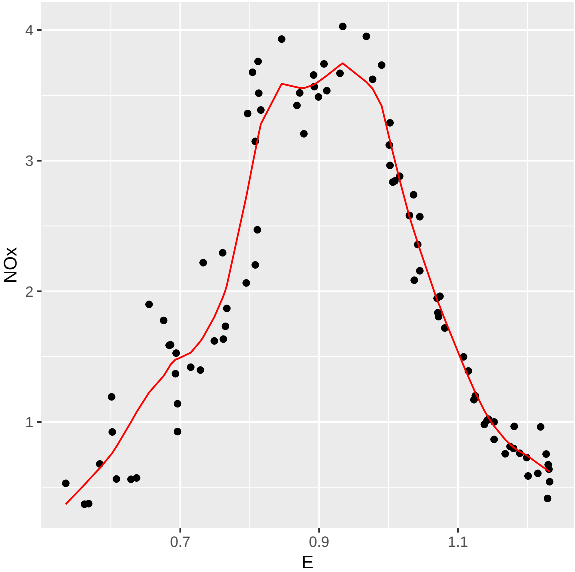
  - β_0^i and β_1^i are the intercept and slope of the linear relation between x and Y in the neighborhood of x_i.

Local regression

```
fit.local.lin.reg = loess(NOx ~ E, data=ethanol,
  degree=1, span=0.19)

df.fit.local.lin.reg = data.frame(E = ethanol$E,
  NOx.fit = fit.local.lin.reg$fitted)
library(dplyr)
p = ggplot() +
  geom_point(data = ethanol, aes(x = E, y=NOx)) +
  geom_line(data = df.fit.local.lin.reg,
    aes(x = E, y = NOx.fit), color = "red")
```

Local regression



Local polynomial regression

- **HWC** comment 8, page 665.
- Use local polynomial regression in place of linear regression.
- Let $w_1^i, \cdots, w_n^i$ be the weights determined by the centered and scaled W for a particular point x_i.
  - The weighted local polynomial regression is found by minimizing

$$\sum_{j=1}^{n} w_j^i \left(Y_i - \beta_0^i - \beta_1^i x_j - \cdots - \beta_d^i x_j^d\right)^2 .$$

- `loess` function in R allows for degrees of $d = 0, 1, 2$.

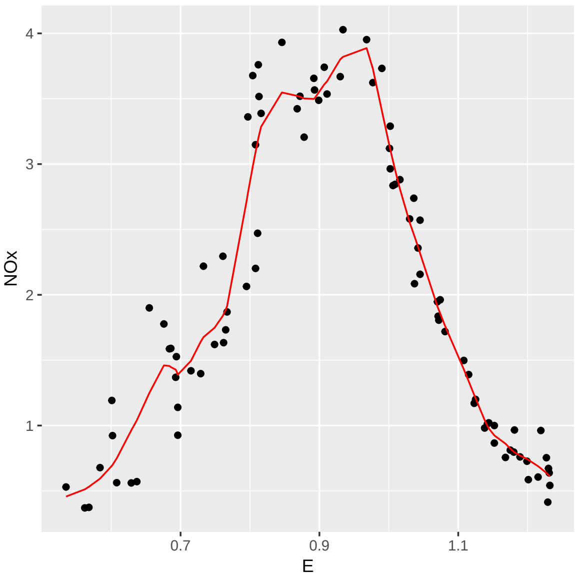
Local polynomial regression

```
fit.local.poly.reg = loess(NOx ~ E, data=ethanol,
  degree=2, span=0.2)

df.fit.local.poly.reg = data.frame(E = ethanol$E,
  NOx.fit = fit.local.poly.reg$fitted)

p = ggplot() +
  geom_point(data = ethanol, aes(x = E, y=NOx)) +
  geom_line(data = df.fit.local.poly.reg,
    aes(x = E, y = NOx.fit), color = "red")
```

Local polynomial regression



Kernel smoothing

- **HWC** Chapter 14.3.
- This is not a nearest neighbor method for a given kernel K and bandwidth h.
  - The number of observations used in the estimate at any point x is not fixed but the window size is.
  - The bandwidth h is the changing value over which the least squares are minimized, rather than the span.
  - Weights are determined by how close each observation x_j to point x, bandwidth, and the kernel K.
- `npreg` from package `np`.
  - `npregbw` from package `np` for bandwidth selection.

Kernel smoothing

- **W 2006** Chapter 5.4.
- Let $h > 0$ - bandwidth.
- Nadaraya (1964, 1965) and Watson (1964):

$$\hat{r}_n(x) = \sum_{i=1}^{n} l_i(x)\, Y_i, \tag{2}$$

where K is a kernel and

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

- The local average regression in (1) is a kernel estimator based on the boxcar kernel.
- We can show that kernel smoother is a linear smoother as in (2).

Kernel smoothing

- The choice of kernel K is not too important.
- Risk is sensitive for h_n which controls the amount of smoothing and depends on sample size n.
  - Small h_n gives rough estimates.
  - Larger h_n's give smoother estimates.

Kernel smoothing

- An example to show the bandwidth affects the estimate.
- Let $x_1, x_2, \cdots, x_n$ be random draws from some density f.
- The risk (integrated squared error loss) of the Nadaraya-Watson kernel estimator is

$$R\left(\hat{r}_n, r\right) = \frac{h_n^4}{4}\left(\int x^2 K\left(x\right) dx\right)^2 \int \left(r^{''}\left(x\right) + 2r^{'}\left(x\right)\frac{f^{'}\left(x\right)}{f\left(x\right)}\right)^2 dx + \frac{\sigma^2 \int K^2\left(x\right) dx}{nh_n}\int \frac{1}{f\left(x\right)} dx + o\left(nh_n^{-1}\right) + o\left(h_n^4\right) \tag{3}$$

as $h_n \to 0$ and $nh_n \to \infty$.

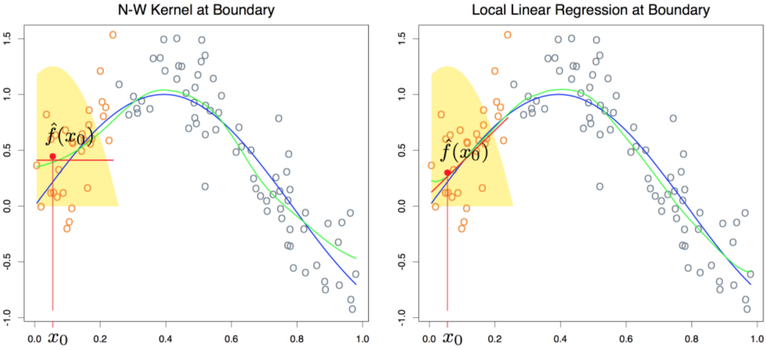
- **Design bias**: $2r^{'}\left(x\right)\frac{f^{'}\left(x\right)}{f\left(x\right)}$ The bias term in (3) depends on the distribution of x_i's.

Kernel smoothing

- The optimal bandwidth will depend on the unknown function r. So we can use cross-validation to find the optimal bandwidth h^*.

Kernel smoothing

- Kernel estimators have high bias near the boundaries called **boundary bias**.



Source: Hastie, Tibshirani, Friedman (2009)

Kernel smoothing

- Alleviate the boundary bias and design bias using local polynomial regression.
  - Use the kernel K as the weight in the local polynomial regression.
  - Estimate is a linear smoother.

Kernel smoothing

```
library(np)
ethanol.npreg <- npreg(bws=.09,
  txdat=ethanol$E,
     tydat=ethanol$NOx,
  ckertype="epanechnikov")
ethanol.npreg2 <- npreg(bws=.03,
  txdat=ethanol$E,
     tydat=ethanol$NOx,
  ckertype="epanechnikov")
ethanol.npreg$MSE
```

```
## [1] 0.2816102
```

```
ethanol.npreg2$MSE
```

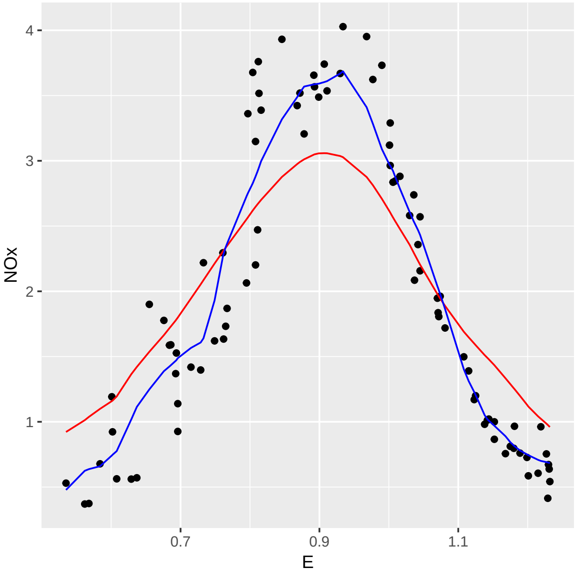
```
## [1] 0.1060051
```

Kernel smoothing

```r
ethanol.npreg.fit = data.frame(E = ethanol$E,
  NOx = ethanol$NOx,
  kernel.fit = fitted(ethanol.npreg),
  kernel.fit2 = fitted(ethanol.npreg2))

ggplot(ethanol.npreg.fit) +
  geom_point(aes(x = E, y = NOx)) +
  geom_line(aes(x = E, y= kernel.fit), color = "red") +
  geom_line(aes(x = E, y= kernel.fit2), color = "blue")
```

Kernel smoothing



Penalized regression

- **W2006** Chapter 5.5
- $Y_i = r(x_i) + \epsilon_i$.
- Suppose we estimate r by choosing $\hat{r}_n(x)$ to minimize the sum of squares
$$\sum_{i=1}^{n} (Y_i - \hat{r}_n(x))^2 .$$
  - Minimizing over all linear functions gives least squares estimator.
  - Minimizing over all functions yields a function that interpolate the data.
- To avoid the above two extreme solutions
  - locally weighted sums of squares (local averages, local linear/polynomial regression, kernel smoother).
  - minimize the penalized sums of squares.

Penalized regression

- Compute $\hat{r}_n$ by minimizing penalized sums of squares

$$M(\lambda) = \sum_i \left(Y_i - \hat{r}_n(x_i)\right)^2 + \lambda J(r),$$

where

$$J(r) = \int \left(r^{''}(x)^2 \, dx\right).$$

- When $\lambda = 0$, the solution is interpolating function.
- When $\lambda \to \infty$, $\hat{r}_n$ converges to the least squares line.
- What does $\hat{r}_n$ looks like for $0 < \lambda < \infty$?

Splines

- A spline is a special piece-wise polynomial.
- A cubic spline
  - Let $\zeta_1, \zeta_2, \cdots, \zeta_k$ be a set of ordered points - called knots - contained in some interval (a, b).
  - A cubic spline is a continuous function r such that (i) r is a cubic polynomial over $(\zeta_1, \zeta_2), \cdots$ and (ii) r has first and second derivatives at knots.

Smoothing splines

- The function $\hat{r}_n(x)$ that minimizes $M(\lambda)$ with penalty $J(r)$ is a natural cubic spline with knots at the data points.
  - $\hat{r}_n$ does not have an explicit form.
  - Smoothing splines.
- Build an explicit basis using B-splines

$$\hat{r}_n(x) = \sum_{j=1}^{N} \hat{\beta}_j B_j(x),$$

  - where $B_1, \cdots, B_N$ are a basis for B-splines with $N = n + 4$.
  - Now we only need to find the coefficients $\hat{\boldsymbol{\beta}} = \left(\hat{\beta}_1, \cdots, \hat{\beta}_N\right)^T$.

B-Splines

- By expanding r in the basis we can now rewrite the minimization as follows:

$$\text{minimize} \left(Y - \mathbf{B}\beta\right)^T \left(Y - \mathbf{B}\beta\right) + \lambda\beta^T\Omega\beta,$$

where $\mathbf{B}_{ij} = B_j\left(X_i\right)$ and $\Omega_{ij} = \int B_j''\left(x\right)B_k''\left(x\right)dx$.

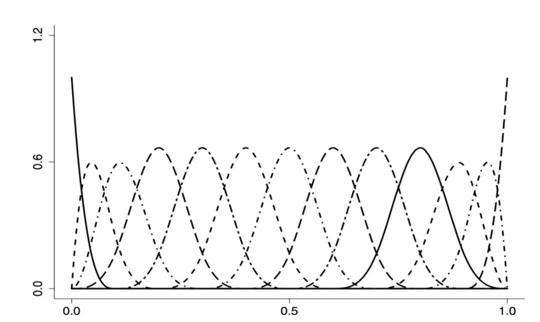
- $$\hat{\beta} = \left(\mathbf{B}^T\mathbf{B} + \lambda\Omega\right)^{-1}\mathbf{B}^T Y.$$

- The smoothing spline is a linear smoother:

$$\boldsymbol{r} = \left(\mathbf{B}^T\mathbf{B} + \lambda\Omega\right)^{-1}\mathbf{B}^T\boldsymbol{Y} = \mathbf{L}\boldsymbol{Y}.$$

B-Splines

- Cubic B-spline basis using nine equally spaced knots on (0,1).



Source: Wasserman (2006)

Splines (Example)

```r
library(splines)
```

- A Cubic Spline with 3 Knots

```r
range(ethanol$E)
```

```
## [1] 0.535 1.232
```

```r
cubic.spline.fit = lm(NOx ~ bs(E,
  knots = c(.75,1,1.2)),
  data = ethanol)
```

Splines (Example)

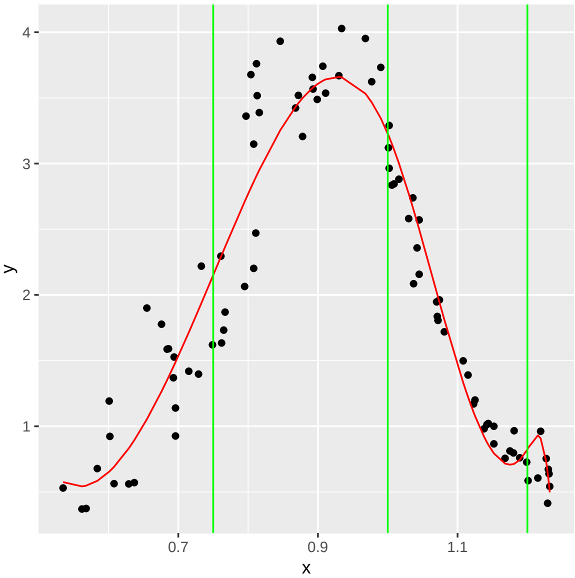
```r
summary(cubic.spline.fit)
```

```
##
## Call:
## lm(formula = NOx ~ bs(E, knots = c(0.75, 1, 1.2)), data
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6686 -0.2063  0.0214  0.1579  0.8616
##
## Coefficients:
##                                  Estimate Std. Error t va
## (Intercept)                       0.57488    0.26526   2.
## bs(E, knots = c(0.75, 1, 1.2))1 -0.22004    0.45474  -0.
## bs(E, knots = c(0.75, 1, 1.2))2  1.70393    0.30013   5.
## bs(E, knots = c(0.75, 1, 1.2))3  4.56864    0.39930  11.
## bs(E, knots = c(0.75, 1, 1.2))4 -0.75105    0.32717  -2.
## bs(E, knots = c(0.75, 1, 1.2))5  0.53076    0.35427   1.
```

Splines (Example)

```
df.cubic.spline = data.frame(x = ethanol$E,
  y = ethanol$NOx,
  fit = fitted(cubic.spline.fit))
p = ggplot(data = df.cubic.spline) +
  geom_point(aes(x = x, y = y)) +
  geom_line(aes(x = x, y = fit),
    color = "red") +
  geom_vline(xintercept = c(.75,1,1.2),
    color = "green")
```

Splines (Example)



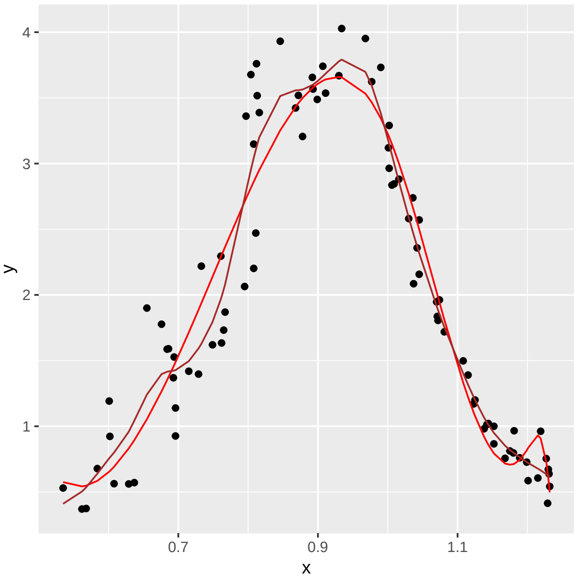
Smoothing spline (Example)

```r
smooth.spline.fit =smooth.spline(ethanol$E,
  ethanol$NOx, cv = TRUE)

df.smooth.splines = data.frame(x = smooth.spline.fit$x,
  fit.smooth.spline = smooth.spline.fit$y)

p = ggplot() +
  geom_point(data = df.cubic.spline,
    aes(x = x, y = y)) +
   geom_line(data = df.cubic.spline,
     aes(x = x, y = fit), color = "red") +
    geom_line(data = df.smooth.splines,
      aes(x = x, y = fit.smooth.spline),
      color = "brown")
```

Smoothing spline (Example)



References for this lecture

HWC Chapter 14 (smoothing)

W Chapter 5