Introduction to Nonparametric Regression

Nathaniel E. Helwig

Assistant Professor of Psychology and Statistics
University of Minnesota (Twin Cities)

Updated 04-Jan-2017

Copyright

Outline of Notes

1) Need for NP Reg
- Motivating example
- Nonparametric regression

2) Local Averaging
- Overview
- Examples

3) Local Regression:
- Overview
- Examples

4) Kernel Smoothing:
- Overview
- Examples

Need for Nonparametric Regression

Results From Four Hypothetical Studies

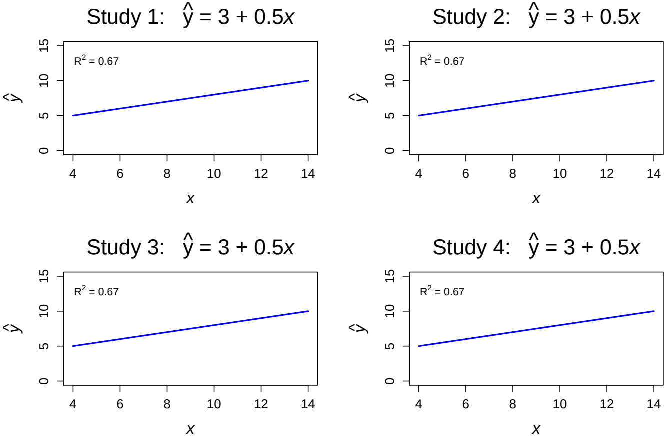


Figure: Estimated linear relationship from four hypothetical studies.

Implications of Four Hypothetical Studies

What do the results on the previous slide imply?

Can we conclude that there is a linear relationship between X and Y?

Is the reproducibility of the finding indicative of a valid discovery?

What have we learned about the data from these results?

Let's Look at the Data

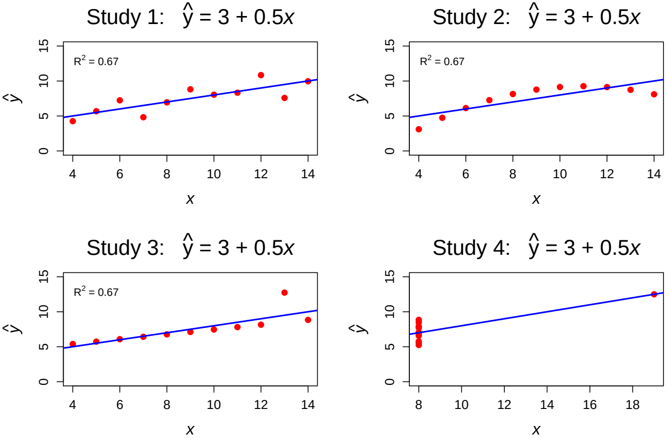


Figure: Estimated linear relationship with corresponding data.

Anscombe's (1973) Quartet

```
> anscombe
   x1 x2 x3 x4    y1   y2    y3    y4
1  10 10 10  8  8.04 9.14  7.46  6.58
2   8  8  8  8  6.95 8.14  6.77  5.76
3  13 13 13  8  7.58 8.74 12.74  7.71
4   9  9  9  8  8.81 8.77  7.11  8.84
5  11 11 11  8  8.33 9.26  7.81  8.47
6  14 14 14  8  9.96 8.10  8.84  7.04
7   6  6  6  8  7.24 6.13  6.08  5.25
8   4  4  4 19  4.26 3.10  5.39 12.50
9  12 12 12  8 10.84 9.13  8.15  5.56
10  7  7  7  8  4.82 7.26  6.42  7.91
11  5  5  5  8  5.68 4.74  5.73  6.89
```

Parametric versus Nonparametric Regression

The general linear model is a form of parametric regression, where the relationship between X and Y has some predetermined form.

- Parameterizes relationship between X and Y, e.g., $\hat{Y} = \beta_0 + \beta_1 X$
- Then estimates the specified parameters, e.g., β_0 and β_1
- Great if you know the form of the relationship (e.g., linear)

In contrast, nonparametric regression tries to estimate the form of the relationship between X and Y.

- No predetermined form for relationship between X and Y
- Great for discovering relationships and building prediction models

Problem of Interest

Smoothers (aka nonparametric regression) try to estimate functions from noisy data.

Suppose we have n pairs of points (x_i, y_i) for $i \in \{1, \ldots, n\}$, and WLOG assume that $x_1 \leq x_2 \leq \cdots \leq x_n$.

Also, suppose the following assumptions hold:

(A1) There is a functional relationship between x and y of the form

$$y_i = \eta(x_i) + \epsilon_i; \quad i \in \{1, \ldots, n\}$$

(A2) The ϵ_i are iid from some distribution $f(x)$ with zero mean

Local Averaging

Friedman's (1984) Local Averaging

To estimate η at the point x_i, we could calculate the average of the y_j values corresponding to x_j values that are "near" x_i.

Friedman (1984) defined "near" as the smallest symmetric window around x_i that contains s observations.

- Note that s is called the *span*
- Size of averaging window can differ for each x_i
- But always use s points in averaging window

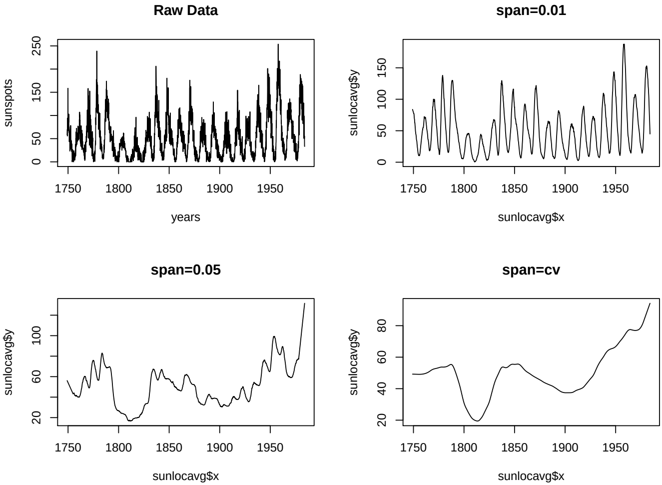
Selecting the Span

Friedman proposed using a cross-validation approach to select span s.

For a given span s, leave-one-out cross-validation:

- Let $\hat{y}_{(i)}$ denote the local averaging estimate of η at the point x_i obtained by holding out the i-th pair (x_i, y_i)
- Define CV residuals $e_i(s) = y_i - \hat{y}_{(i)}$; note residual is function of s
- $\hat{s} = \min_{s \in \mathcal{S}} (1/n) \sum_{i=1}^{n} e_i^2(s)$

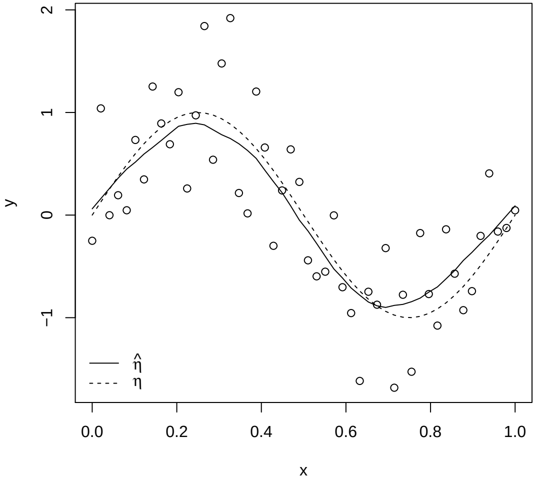
Local Averaging Example 1: `sunspots` data



Local Averaging Example 1: R code

```
data(sunspots)
yrs=start(sunspots)
yre=end(sunspots)
years=seq(yrs[1]+yrs[2]/12,yre[1]+yre[2]/12,by=1/12)
dev.new(width=8,height=6,noRStudioGD=TRUE)
par(mfrow=c(2,2))
plot(years,sunspots,type="l",main="Raw Data")
sunlocavg=supsmu(years,sunspots,span=0.01)
plot(sunlocavg,type="l",main="span=0.01")
sunlocavg=supsmu(years,sunspots,span=0.05)
plot(sunlocavg,type="l",main="span=0.05")
sunlocavg=supsmu(years,sunspots)
plot(sunlocavg,type="l",main="span=cv")
```

Local Averaging Example 2: simulated data



Local Averaging Example 2: R code

```
> set.seed(1)
> x=seq(0,1,length=50)
> y=sin(2*pi*x)+rnorm(50,sd=0.5)
> locavg=supsmu(x,y)
> dev.new(width=6,height=6,noRStudioGD=TRUE)
> plot(x,y)
> lines(locavg$x,locavg$y)
> lines(locavg$x,sin(2*pi*x),lty=2)
> legend("bottomleft",c(expression(hat(eta)),
+                       expression(eta)),lty=1:2,bty="n")
```

Local Regression

Cleveland's (1979) Local Regression

To estimate η at the point x_i, we could calculate the local linear regression line using the (x_j, y_j) points "near" x_i.

- LOWESS: LOcally WEighted Scatterplot Smoothing
- LOESS: LOcal regrESSion

Cleveland (1979) proposed using weighted regression with weights related to distance of x_j points to x_i.

- Weight function is scaled so only proportion of (x_j, y_j) points are used in each regression
- Size of regression window can differ for each x_i
- But use αn points in each regression where $\alpha \in (0, 1]$

Weighted Local Regression

Cleveland proposed using a weight function W such that

$$W(x) \begin{cases} > 0, & |x| < 1 \\ = 0, & |x| \geq 1 \end{cases}$$

Then W is modified for each index $i \in \{1, \ldots, n\}$ by...

- Centering W at x_i
- Scaling W such that αn values are nonzero

R's `loess` uses tricube function: $W(x) = (1 - |x|^3)^3$ for $|x| < 1$

Weighted Local Regression (continued)

Let $\{w_j^i\}_{j=1}^n$ denote the weights for a particular point x_i.

The weighted local regression problem minimizes

$$\sum_{j=1}^{n} w_j^i (y_j - \beta_0^i - \beta_1^i x_j)^2$$

where β_0^i and β_1^i are intercept and slope between x and y in neighborhood around x_i

Robust Weighted Local Regression

To reduce effect of outliers, we can perform another regression with weights based on the residuals $\hat{\epsilon}_i = y_i - \hat{y}_i$ where $\hat{y}_i = \hat{\beta}_0^i + \hat{\beta}_1^i x_i$.

- Bisquare weight function: $B(x) = (1 - x^2)^2$, $|x| < 1$
- Residual-based weights: $\delta_i = B\left(\hat{\epsilon}_i / (6\text{median}_{1 \leq j \leq n} |\hat{\epsilon}_j|)\right)$

The robust weighted local regression problem minimizes

$$\sum_{j=1}^{n} \delta_j w_j^i (y_j - \beta_0^{i'} - \beta_1^{i'} x_j)^2$$

where $\beta_0^{i'}$ and $\beta_1^{i'}$ are the robust (i.e., residual corrected) intercept and slope between x and y in neighborhood around x_i

Selecting the Span

Want to minimize the leave-one-out cross-validation criterion:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_{(i)})^2$$

where $\hat{y}_{(i)}$ is the LOESS estimate of y_i obtained by holding out (x_i, y_i).

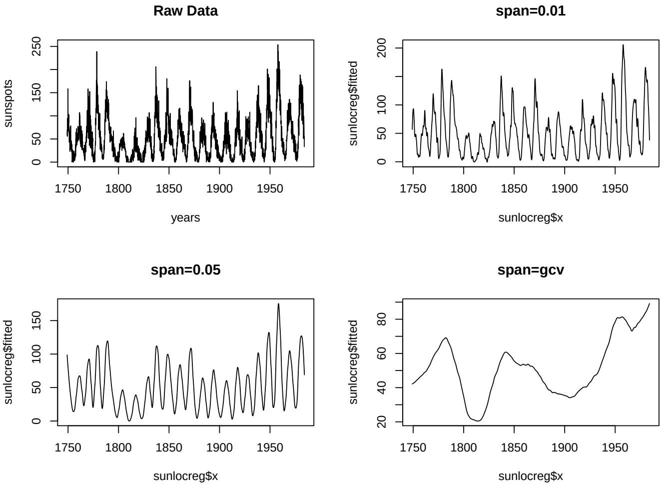
Rewrite the leave-one-out cross-validation criterion as

$$\frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{(1 - h_{ii})^2}$$

where h_{ii} are diagonal entries of the hat matrix $\mathbf{H}$ that determines $\hat{y}_i$.

- Replace h_{ii} with $\frac{1}{n}\sum_{i=1}^{n} h_{ii} = \mathrm{tr}(\mathbf{H})/n$ for generalized CV

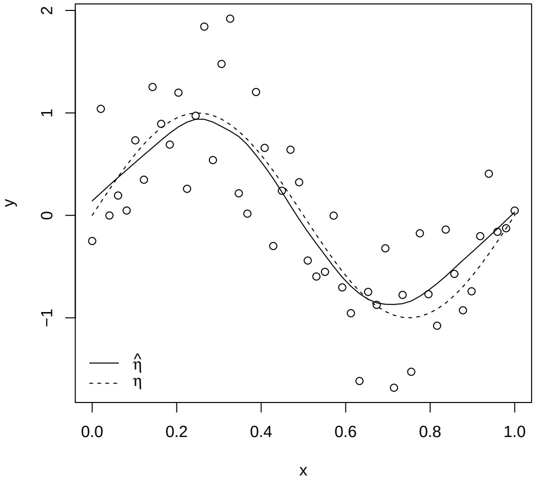
Local Regression Example 1: `sunspots` data



Local Regression Example 1: R code

```
library(fANCOVA)
data(sunspots)
yrs=start(sunspots)
yre=end(sunspots)
years=seq(yrs[1]+yrs[2]/12,yre[1]+yre[2]/12,by=1/12)
dev.new(width=8,height=6,noRStudioGD=TRUE)
par(mfrow=c(2,2))
plot(years,sunspots,type="l",main="Raw Data")
sunlocreg=loess(sunspots~years,span=0.01)
plot(sunlocreg$x,sunlocreg$fitted,type="l",main="span=0.01")
sunlocreg=loess(sunspots~years,span=0.05)
plot(sunlocreg$x,sunlocreg$fitted,type="l",main="span=0.05")
sunlocreg=loess.as(years,sunspots,criterion="gcv")
plot(sunlocreg$x,sunlocreg$fitted,type="l",main="span=gcv")
```

Local Regression Example 2: simulated data



Local Regression Example 2: R code

```
> library(fANCOVA)
> set.seed(55455)
> x=seq(0,1,length=50)
> y=sin(2*pi*x)+rnorm(50,sd=0.5)
> locreg=loess.as(x,y)
> dev.new(width=6,height=6,noRStudioGD=TRUE)
> plot(x,y)
> lines(locreg$x,locreg$fitted)
> lines(locreg$x,sin(2*pi*x),lty=2)
> legend("bottomleft",c(expression(hat(eta)),
+                       expression(eta)),lty=1:2,bty="n")
```

Kernel Smoothing

Kernel Smoothing for General Functions

Kernel smoothing extends KDE idea to estimation a general function η.

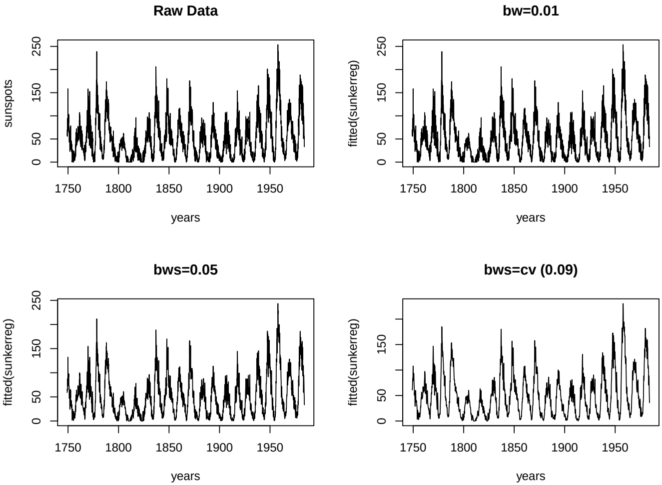
Nadaraya (1964) and Watson (1964) independently introduced the kernel regression estimate

$$\hat{\eta}(x) = \frac{\sum_{i=1}^{n} y_i K\left(\frac{x-x_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right)} = \sum_{i=1}^{n} y_i w_i$$

where weights $w_i = \frac{K\left(\frac{x-x_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right)}$ are dependent on chosen K and h.

CV, GCV, or AIC to select bandwidth in kernel regression problems.

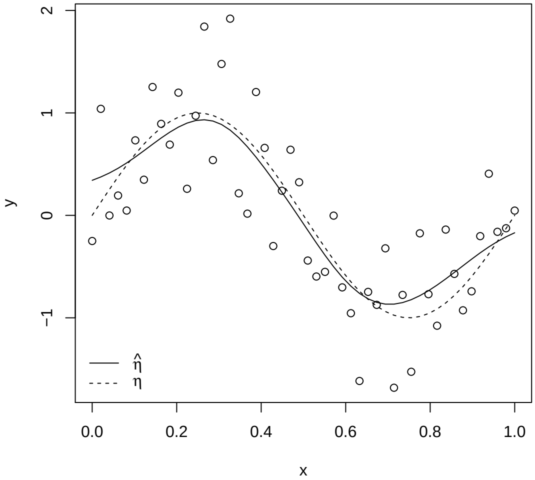
Kernel Smoothing Example 1: `sunspots` data



Kernel Smoothing Example 1: R code

```
library(np)
data(sunspots)
yrs=start(sunspots)
yre=end(sunspots)
sunspots=as.vector(sunspots)
years=seq(yrs[1]+yrs[2]/12,yre[1]+yre[2]/12,by=1/12)
dev.new(width=8,height=6,noRStudioGD=TRUE)
par(mfrow=c(2,2))
plot(years,sunspots,type="l",main="Raw Data")
sunkerreg=npreg(bws=0.01,txdat=years,tydat=sunspots)
plot(years,fitted(sunkerreg),type="l",main="bw=0.01")
sunkerreg=npreg(bws=0.05,txdat=years,tydat=sunspots)
plot(years,fitted(sunkerreg),type="l",main="bws=0.05")
sunkerreg=npreg(txdat=years,tydat=sunspots)
plot(years,fitted(sunkerreg),type="l",main="bws=cv (0.09)")
```

Kernel Smoothing Example 2: simulated data



Kernel Smoothing Example 2: R code

```
> library(np)
> set.seed(55455)
> x=seq(0,1,length=50)
> y=sin(2*pi*x)+rnorm(50,sd=0.5)
> kerreg=npreg(txdat=x,tydat=y)

> dev.new(width=6,height=6,noRStudioGD=TRUE)
> plot(x,y)
> lines(x,fitted(kerreg))
> lines(x,sin(2*pi*x),lty=2)
> legend("bottomleft",c(expression(hat(eta)),
+                       expression(eta)),lty=1:2,bty="n")
```