Administration

- HW 2 posted on web page, due March 4 by 1 pm
- Midterm on March 16; practice questions coming
- Lecture/questions on Thursday this week
- Regression: variable selection, regression splines, smoothing splines, wavelet smoothing
- Classification: discriminant analysis, logistic regression
- Kernel Smoothing Methods; Model Assessment and Selection
- Projection Pursuit Regression and Neural Networks, Ch. 11
- Support Vector Machines, Ch. 12
- Classification and Regression Trees, Ch. 9.2
- Unsupervised Learning, Ch. 14

Wavelet examples

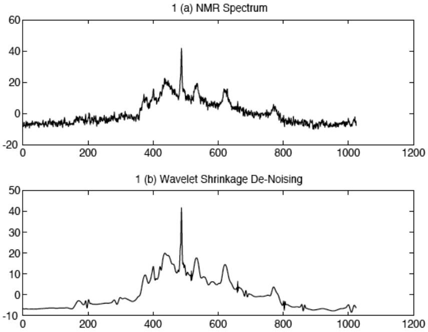
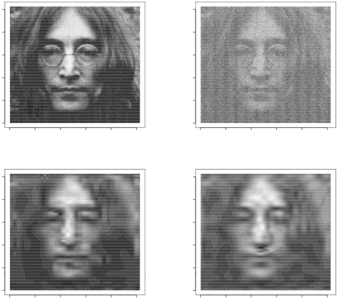


Figure 1: First Figure of Short Course

Buckheit et al., "About Wavelab" 2005 From
`http://www-stat.stanford.edu/~wavelab/Wavelab_850/Documentation.html`
Compare Figure 5.17

... wavelets



Vidaković and Müller, ”Wavelets for kids (Part I)” 1994. From `http://www.amara.com/current/wavelet.html`: ”Amara's wavelet page”

```
> library(wavethresh); data(lennon)
```

Ch. 6: Kernel smoothing methods – smoothing without basis functions

- model: $E(Y \mid x) = f(x)$ ("smooth")
- data: $\quad y_i = f(x_i) + \epsilon_i$
- simplest possible estimate of $f(x_0) = E(Y \mid x_0)$:
- $\hat{f}(x_0) = \mathrm{ave}(y_i \mid x_i \in N_k(x_0))$ running means
- $N_k(x_0)$ set of k smallest values of $|x_i - x_0|$ nearest neighbours
- weight cases according to distance from x_0

$$\hat{f}(x_0) = \frac{\sum_{i=1}^{N} K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)} \quad (6.2)$$

Figure 6.1

- kernel function

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{\lambda}\right) \text{ or } D\left(\frac{|x - x_0|}{h_\lambda(x_0)}\right)$$

... kernel smoothing

- λ determines the width of the neighbourhood, hence smoothness
- increasing λ gives smoother function (higher bias, lower variance)
- constant (metric) window width –
  constant bias, variance $\propto 1/\text{local density}$
- nearest neighbour window width $h_\lambda(x_0)$ –
  constant variance, bias $\propto 1/\text{local density}$
- Choice of kernel:

$$
\begin{aligned}
D(t) &= \begin{cases} \frac{3}{4}(1-t^2), |t| \leq 1 & \text{Epanichakov} \\ 0 & \end{cases} \\
&= \begin{cases} (1-|t|^3)^3, |t| \leq 1 & \text{tri}-\text{cube} \\ 0 & \end{cases} \\
&= \phi(t) = \frac{1}{\sqrt{2\pi}} \exp(-t^2/2) \qquad \text{Gaussian}
\end{aligned}
$$

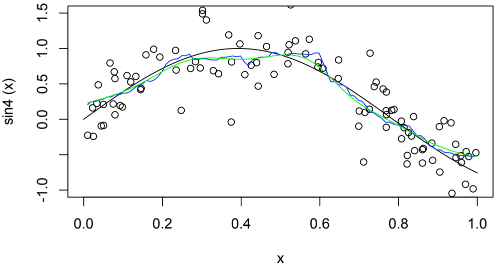
- could add weights w_i to each observation (p.194)

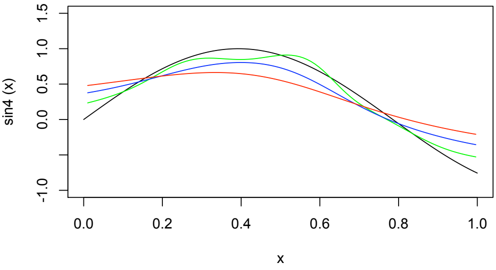
R:

```
ksmooth(x,y,kernel=c("box","normal"),bandwidth=0.5,
         range.x=range(x),
         n.points=max(100,length(x)), x.points)

> eps<-rnorm(100,0,1/3)
> x<-runif(100)
> sin4 <- function(x){sin(4*x)}
> y<-sin4(x)+eps
> plot(sin4,0,1,type="l",ylim=c(-1.0,1.5),xlim=c(0,1))
> points(x,y)
> lines(ksmooth(x,y,"box",bandwidth=.2),col="blue")
> lines(ksmooth(x,y,"normal",bandwidth=.2),col="green")
> plot(sin4,0,1,type="l",ylim=c(-1.0,1.5),xlim=c(0,1))
> lines(ksmooth(x,y,"normal",bandwidth=.2),col="green")
> lines(ksmooth(x,y,"normal",bandwidth=0.4),col="blue")
> lines(ksmooth(x,y,"normal",bandwidth=0.6),col="red")
```

(Figure 6.1)


sin4 (x)
-1.0
0.0
0.5
1.0
1.5
0.0
0.2
0.4
0.6
0.8
1.0
x


sin4 (x)
-1.0
0.0
0.5
1.0
1.5
0.0
0.2
0.4
0.6
0.8
1.0
x

Local linear regression (§6.6.1)

- replace weighted average of x_i's with weighted linear (or polynomial) regression: better endpoint behaviour
- $$\min_{\alpha(x_0),\beta(x_0)} \sum K_\lambda(x_0, x_i)\{y_i - \alpha(x_0) - \beta(x_0)x_i\}^2$$
- $$\hat{f}(x_0) = (1, x_0)(X^T W(x_0) X)^{-1} X^T W(x_0) y$$
- $$X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix} = B$$
- $W(x_0)_{ii} = K_\lambda(x_0, x_i), \quad W(x_0)_{ij} = 0$

Notes

- Recall weighted least squares:

$$\min_\beta \sum w_i(y_i - \beta_0 - \beta_1 x_i)^2 \text{or} \min_\beta (y - X\beta)^T W(y - X\beta)$$

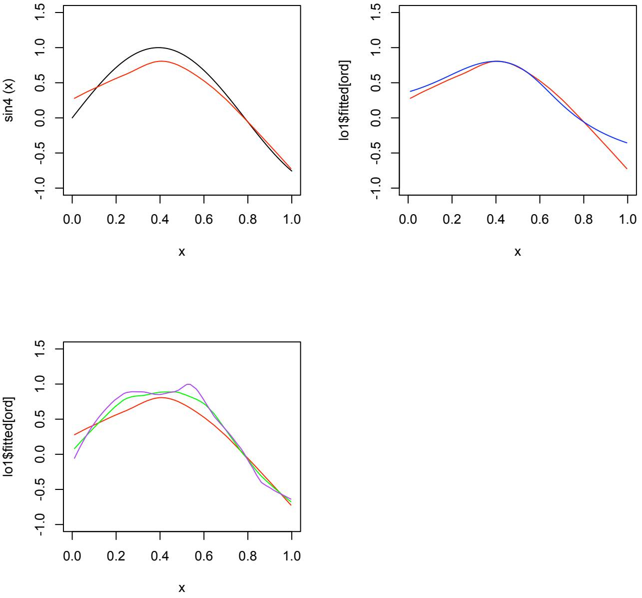
- $$\hat{\beta} = (X^T WX)^{-1} X^T Wy$$
- can combine the least squares weights with the kernel weights Figure 6.4, pp. 196
- can also do local quadratic regression (and higher) but increases bias at endpoints
- for extrapolation book recommends local linear fits; for good fits in middle local quadratic
- In R there are several smoothers: `ksmooth` and `loess` are built in
- The first uses kernel smoothing, the second uses local linear regression (robustified)

- `scatter.smooth` fits a `loess` curve to a scatter plot
- `loess` takes a `family` argument : `family = gaussian` gives weighted least squares using K_λ as weights and `family=symmetric` gives a robust version using Tukey's biweight
- `supsmu` implements "Friedman's super smoother": a running lines smoother with elaborate adaptive choice of bandwidth
- Library `KernSmooth` has `locpoly` for local polynomial fits, and by setting `degree = 0` gives a kernel smooth
- as usual more smoothing means larger bias, smaller variance

```
 ## file loess.R contains the following lines:
plot(sin4,0,1,type="l",ylim=c(-1,1.5),xlim=c(0,1), xlab = "x")
lo1 = loess(y ~ x, degree = 1, span = 0.75)
## we are using data generated in ksmooth
attributes(lo1)
ord = order(lo1$x)
lines(lo1$x[ord],lo1$fitted[ord],col="red")
plot(lo1$x[ord],lo1$fitted[ord],type="l",col="red",
ylim=c(-1,1.5),xlim=c(0,1), xlab = "x")
lines(ksmooth(x,y,"normal",bandwidth=0.4),col="blue")
lo2 = loess(y~x, degree=1, span=0.4)
lo3 = loess(y~x, degree=2, span=0.4)
plot(lo1$x[ord],lo1$fitted[ord],type="l",col="red",
  ylim=c(-1,1.5),xlim=c(0,1), xlab = "x")
lines(lo1$x[ord],lo2$fitted[ord],col="green")
lines(lo1$x[ord],lo3$fitted[ord],col="purple")
## end file
## I ran these commands using source("loess.R"), or the File Menu
## After making sure the file was in the same directory that I was working in

> attributes(lo1)
$names
 [1] "n"          "fitted"     "residuals" "enp"        "s"          "one.delta"
 [7] "two.delta" "trace.hat" "divisor"   "pars"       "kd"         "call"
[13] "terms"      "xnames"     "x"          "y"          "weights"

$class
[1] "loess"
```


sin4 (x)
lo1$fitted[ord]
lo1$fitted[ord]
x
x
x
-1.0 -0.5 0.0 0.5 1.0 1.5
0.0 0.2 0.4 0.6 0.8 1.0

Notes

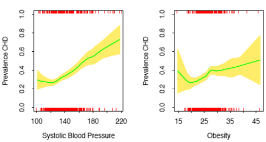
- $\hat{f} = S_\lambda y$ and df=trace(S_λ), as in smoothing splines p.199
- while possible to fit these models in R^p, §6.3, 6.4, doesn't seem so useful Figure 6.8
- §6.4 describes ways to impose some structure to get a more interpretable model
- can use the same kernel smoothing idea for likelihood functions and maximum likelihood estimates:

$$\max_\beta \sum \ell(\beta; y_i)$$

replaced by

$$\max_\beta \sum K_\lambda(x_0, x_i)\ell(\beta; y_i)$$

called local likelihood and described in §6.5 Figure 6.12



Kernel methods for classification (§6.6)

- estimate density of predictor from sample $x_1, \ldots, x_N$
- rather than assume normality as in LDA
- $\hat{f}(x_0) = \dfrac{\#\{x_i \in \mathcal{N}_\lambda(x_0)\}}{N\lambda}$
- histogram if intervals don't overlap
- otherwise a bumpy density estimate
- use kernel to smooth as before
- $\hat{f}(x_0) = \frac{1}{N\lambda} \sum K_\lambda(x_0, x_i)$: smooth density estimate
- implemented in `R` as `density(x, ...)` with a large choice of kernels; default is Gaussian
- $$\hat{f}(x_0) = \frac{1}{N} \sum_{i=1}^{N} \phi_\lambda(x_0 - x_i) = (\hat{F} \star \phi_\lambda)(x_0)$$
- $$\hat{F}(x) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\{x_i \leq x\}$$

... kernel methods (§6.6.2)

- for classification: compute $\hat{f}_j(X)$ for each class

$$\hat{\text{pr}}(Y = j \mid X = x_0) = \hat{\pi}_j \hat{f}_j(x_0) / \sum \hat{\pi}_k \hat{f}_j(x_0)$$

- with p inputs treat the inputs as independent
- $$\hat{f}_j(X) = \Pi_{k=1}^{p} \hat{f}_{jk}(X_k)$$
- Naive Bayes classifier (§6.6.3):

$$\hat{\text{pr}}(Y = j \mid X = x_0) = \frac{\hat{\pi}_j \hat{f}_j(x_0)}{\Sigma \hat{\pi}_j \hat{f}_j(x_0)}$$

- Figure 6.15

Which smoothing method?

- basis functions: natural splines, Fourier, wavelet bases
- regularization
- cubic smoothing splines
- kernel smoothers: locally constant/linear/polynomial
- adaptive bandwidth, running medians, running *M*-estimates
- Dantzig selector, elastic net, rodeo (Lafferty & Wasserman, 2008)
- Faraway (2006) Extending the Linear Model:
  - with very little noise, a small amount of local smoothing (e.g. nearest neighbours)
  - with moderate amounts of noise, kernel and spline methods are effective
  - with large amounts of noise, parametric methods are more attractive
- "It is not reasonable to claim that any one smoother is better than the rest"
  - `loess` is robust to outliers, and provides smooth fits
  - spline smoothers are more efficient, but potentially sensitive to outliers
  - kernel smoothers are very sensitive to bandwidth