Administration

- HW due February 11 by 1 pm
- Thursday, February 4: TA Li Li has office hours 1-2 (in class), 2-3 in SS 6027A
- Chapter 3: §3.1, 3.2 (except 3.2.4), 3.3 (except 3.3.3), 3.4 (except 3.4.4), 3.5.1
- Chapter 4: §4.1, 4.2, 4.3 (except 4.3.1, 4.3.2), 4.4.0, 4.4.1, 4.4.2
- My office hours are Tuesday 3-4 and Thursday 2-3 (although cancelled on Thursday Feb 4)

http://www3.interscience.wiley.com/cgi-bin/fulltext/123233977/HTMLSTART
A few points on logistic regression

- Logistic regression: $Pr(G = k \mid X)$ linear on the logit scale
- Linear discriminant analysis:
  \[ Pr(G = k \mid X) \propto Pr(X \mid G = k)Pr(G = k) \]
- Theory: LDA more efficient if $X$ really is normal
- Practice: LR usually viewed as more robust, but HTF claim prediction errors are very similar
- See: last slide of Jan 26 for calculation of prediction errors on training data
- Data: LR is more complicated with $K > 2$; use `multinom` in the MASS library
- Lasso version of logistic regression described in §4.4.4
... logistic regression

- Deviance in a generalized linear model (such as LR), is 
  $-2 \log L(\hat{\beta}) + constant$

- Comparing deviances from two model fits is a log-likelihood ratio test that the corresponding parameters are 0

- AIC compares instead $-2 \log L(\hat{\beta}) + 2p$

- for Binomial data, but not for binary data, residual deviance provides a test of goodness of fit of the binomial model
Flexible modelling using basis expansions

(Chapter 5)

- Linear regression: \( y = X\beta + \epsilon, \quad \epsilon \sim (0, \sigma^2) \)
- ‘Smooth’ regression: \( y = f(X) + \epsilon \)
- \( f(X) = E(Y \mid X) \) to be specified
- Flexible linear modelling

\[
f(X) = \sum_{m=1}^{M} \beta_m h_m(X)
\]

- This is called a linear basis expansion, and \( h_m \) is the \( m \)th basis function
- For example if \( X \) is one-dimensional:
  \[
  f(X) = \beta_0 + \beta_1 X + \beta_2 X^2, \text{ or }
  f(X) = \beta_0 + \beta_1 \sin(X) + \beta_2 \cos(X), \text{ etc.}
  \]
- Simple linear regression has \( h_1(X) = 1, \ h_2(X) = X \). Several other examples on p.140
Polynomial fits: \( h_j(x) = x^j, j = 0, \ldots, m \)

Fit using linear regression with design matrix \( X \), where \( X_{ij} = h_j(x_i) \)

Justification is that any 'smooth' function can be approximated by a polynomial expansion (Taylor series)

Can be difficult to fit numerically, as correlation between columns can be large

May be useful locally, but less likely to work over the range of \( X \)

Idea: fit polynomials locally in \( X \)

Need to be careful not to overfit, since we are using only a fraction of the data
**FIGURE 5.1.** The top left panel shows a piecewise constant function fit to some artificial data. The broken vertical lines indicate the positions of the two knots $\xi_1$ and $\xi_2$. The blue curve represents the true function, from which the data were generated. The green curve shows the fitted constant function. The top right panel illustrates a piecewise linear function. The bottom left panel depicts a continuous piecewise linear function. The bottom right panel shows a piecewise-linear basis function $$(X - \xi_1)_+$$.
Piecewise polynomials

- piecewise constant basis functions
  \[ h_1(X) = I(X < \xi_1), \quad h_2(X) = I(\xi_1 \leq X < \xi_2), \]
  \[ h_3(x) = I(\xi_2 \leq X) \]
- fitting by local averaging

- piecewise linear basis functions, with constraints
  \[ h_1(X) = 1, \quad h_2(X) = X \]
  \[ h_3(X) = (X - \xi_1)_+, \quad h_4(X) = (X - \xi_2)_+ \]
- windows defined by knots \( \xi_1, \xi_2, \ldots \)
... cubic polynomials

- basis functions
  \[ h_1(X) = 1, \quad h_2(X) = X, \quad h_3(X) = X^2, \quad h_4(X) = X^3 \]

- continuity
  \[ h_5(X) = (X - \xi_1)^3_+, \quad h_6(X) = (X - \xi_2)^3_+ \]

- continuous function, continuous first and second derivatives  Figure 5.2
Cubic splines

- truncated power basis of degree 3
- need to choose number of knots $K$ and placement of knots $\xi_1, \ldots, \xi_K$
- construct features matrix using truncated power basis set
- use constructed matrix as set of predictors

```r
> x = 1:10; cbind(1, x, x^2, x^3); poly(x^3)
> bs(x) ## B-spline basis
```

```
       1         2         3
 [1,] 0.0000000 0.0000000 0.00000000
 [2,] 0.2633745 0.0329218 0.00137174
 [3,] 0.4032922 0.1152263 0.01097394
 [4,] 0.4444444 0.2222222 0.03703704
 [5,] 0.4115226 0.3292181 0.08779149
 ...```
Bone density, Figure 5.6

- `data(bone) in ElemStatLearn`
- 485 observations; 226 male, 259, female
- covariate $x = \text{age}$; response $y = \text{bone density}$
... bone density

```r
> data(bone)  # bs(x) with no other arguments just gives a single cubic polynomial
> bone[1:4,]  # bs(x, df=12) gives a proper cubic spline basis, with 9 knots
   idnum age gender    spnbmd
  1    1 11.70  male  0.018080670
  2    1 12.70  male  0.060109290
  3    1 13.75  male  0.005857545
  4    2 13.25  male  0.010263930
...

> bone.bs.m = with(subset(bone, gender=="male"), lm(spnbmd ~ bs(age)))
> bone.bs.f = with(subset(bone, gender=="female"), lm(spnbmd ~ bs(age)))
> plot(spnbmd ~ age, data = bone, col =
+ ifelse(gender=="male", "blue", "red2"))
> points(bone$age[bone$gender=="male"], bone.bs.m$fitted.values, col="blue", pch="+")
> points(bone$age[bone$gender=="female"], bone.bs.f$fitted.values, col="red", pch="+")
```
... useful code

> bone.bs.f = with(subset(bone,gender=="female"),lm(spnbmd ~ bs(age,df=12)))
> bone.bs.m = with(subset(bone,gender=="male"),lm(spnbmd ~ bs(age,df=12)))
> ordf = order(bone$age[bone$gender=="female"])
> ordm = order(bone$age[bone$gender=="male"])
> plot(spnbmd ~ age, data = bone, col = ifelse(gender=="male", "blue", "red2")
> lines(maleage[ordm],bone.bs.m$fitted.values[ordm],col="blue")
> lines(femage[ordf],bone.bs.f$fitted.values[ordf],col="red")
Heart data

```r
> bs.sb <- bs(hr$sbp, df=4)  # the B-spline basis
> dim(bs.sb)
[1] 462 4  # this is the basis matrix
> bs.sb[1:4,]
    1       2       3       4
[1,] 0.16968090 0.45115900 0.34950621 0.029653925
[2,] 0.35240669 0.47588510 0.17002102 0.001687183
[3,] 0.71090498 0.16424200 0.01087580 0.000000000
[4,] 0.09617733 0.37698080 0.44812470 0.078717201
```

Regression splines (p.144) are linear fits to these basis functions
B-splines

- The $B$-spline basis equivalent to the truncated power basis
- Appendix to Ch. 5 describes the construction
- In R library(splines):
  \[
  \text{bs}(x, \text{df=\texttt{NULL}}, \text{knots=\texttt{NULL}}, \text{degree}=3, \text{intercept=FALSE}, \text{Boundary.knots=range(x)})
  \]
- Must specify either \texttt{df} or \texttt{knots}. For the $B$-spline basis, $\#\text{knots} = \text{df} - \text{degree}$ and degree is usually 3
- Natural cubic splines are linear at the end of the range (§5.2.1)
- \text{n}$s(x, \text{df=\texttt{NULL}}, \text{knots=\texttt{NULL}}, \text{degree}=3, \text{intercept=FALSE}, \text{Boundary.knots=range(x)})$
- For natural cubic splines, $\#\text{knots} = \text{df} - 1$
> heart.ns = glm (chd ~ ns(sbp, 4) + ns(tobacco, 4) + ns(ldl, 4) + famhist + ns(obesity, 4) +
  + ns(age, 4), family=binomial)
> summary(heart.ns)

Call:
glm(formula = chd ~ ns(sbp, 4) + ns(tobacco, 4) + ns(ldl, 4) +
  famhist + ns(obesity, 4) + ns(age, 4), family = binomial)

Deviance Residuals:
  Min       1Q   Median       3Q      Max
-1.7216  -0.8322  -0.3777   0.8870  2.9694

Coefficients:
                                         Estimate Std. Error z value Pr(>|z|)
(Intercept)                              -2.265534   2.367227  -0.957 0.338547
ns(sbp, 4)1                               -1.474172   0.843870  -1.747 0.080652 .
n(sbp, 4)2                               -1.351182   0.759548  -1.779 0.075251 .
n(sbp, 4)3                               -3.729348   2.021064  -1.845 0.065003 .
n(sbp, 4)4                                1.381701   0.995268   1.388 0.165055
ns(tobacco, 4)1                          0.654109   0.453248   1.443 0.148975
ns(tobacco, 4)2                          0.392582   0.892628   0.440 0.660079
ns(tobacco, 4)3                          3.385130   1.179656   2.827 0.004695 **
n(sobp, 4)4                               3.845611   2.386584   1.611 0.107104
famhistPresent                           1.078507   0.237685   4.538 5.69e-06 ***
n(obesity, 4)1                           -3.089393   1.707207  -1.810 0.070355 .
n(obesity, 4)2                           -2.385045   1.200450  -1.987 0.046945 *
n(obesity, 4)3                           -4.998882   3.796264  -1.317 0.187909
ns(obesity, 4)4                          0.009109   1.751127   0.005 0.995850
ns(age, 4)1                              -2.628320   1.116674  -2.354 0.018588 .

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 596.11 on 461 degrees of freedom
Residual deviance: 458.09 on 440 degrees of freedom
AIC: 502.09
Number of Fisher Scoring iterations: 15 / 30
The individual coefficients don’t mean anything, we need to evaluate groups of coefficients. We can do this with successive likelihood ratio tests, by hand, e.g.

```r
> summary(heart.ns)

... stuff omitted

<table>
<thead>
<tr>
<th>ns(age, 4)3</th>
<th>7.624692</th>
<th>2.560613</th>
<th>2.978</th>
<th>0.002904 *</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns(age, 4)4</td>
<td>1.535277</td>
<td>0.591531</td>
<td>2.595</td>
<td>0.009447 **</td>
</tr>
</tbody>
</table>

---

Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 596.11 on 461 degrees of freedom
Residual deviance: 458.09 on 440 degrees of freedom
AIC: 502.09

Number of Fisher Scoring iterations: 6
> update(heart.ns, . ~ . - ns(sbp, 4))

Call: glm(formula = chd ~ ns(tobacco, 4) + ns(ldl, 4) + famhist + ns(obesity, 4) + ns(age, 4), family = binomial)

Coefficients:
  (Intercept) ns(tobacco, 4)1 ns(tobacco, 4)2 ns(tobacco, 4)3 ns(tobacco, 4)4 ns(ldl, 4)1 ns(ldl, 4)2 ns(ldl, 4)3 ns(ldl, 4)4 famhistPresent ns(obesity, 4)1 ns(obesity, 4)2 ns(obesity, 4)3 ns(obesity, 4)4 ns(age, 4)1 ns(age, 4)2 ns(age, 4)3 ns(age, 4)4
-3.91758 0.61696 0.46188 3.51363 3.82464 1.70945 1.70659 4.19515 2.90793 0.99053 -2.93143 -2.32793 -4.87074 -0.01103 2.52772 3.12963 7.34899 1.53433

Degrees of Freedom: 461 Total (i.e. Null); 444 Residual
Null Deviance: 596.1
Residual Deviance: 467.2  AIC: 503.2
> 467.2 - 458.1
[1] 9.1
> pchisq(9.1, df=4)
[1] 0.941352
> 1-.Last.value
[1] 0.05864798  # compare Table 5.1
The function `step` does all this for you:

```r
> step(heart.ns)
Start: AIC=502.09
chd ~ ns(sbp, 4) + ns(tobacco, 4) + ns(ldl, 4) + famhist + ns(obesity, 4) + ns(age, 4)

Df Deviance AIC
<none> 458.09 502.09
- ns(obesity, 4) 4 466.24 502.24
- ns(sbp, 4) 4 467.16 503.16
- ns(tobacco, 4) 4 470.48 506.48
- ns(ldl, 4) 4 472.39 508.39
- ns(age, 4) 4 481.86 517.86
- famhist 1 479.44 521.44
...
> anova(heart.ns)
Analysis of Deviance Table

Model: binomial, link: logit

Response: chd

Terms added sequentially (first to last)

<table>
<thead>
<tr>
<th>Df</th>
<th>Deviance</th>
<th>Resid. Df</th>
<th>Resid. Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>461</td>
<td>596.11</td>
<td>576.85</td>
</tr>
<tr>
<td>ns(sbp, 4)</td>
<td>4</td>
<td>19.26</td>
<td>457</td>
</tr>
<tr>
<td>ns(tobacco, 4)</td>
<td>4</td>
<td>46.90</td>
<td>453</td>
</tr>
<tr>
<td>ns(ldl, 4)</td>
<td>4</td>
<td>19.08</td>
<td>449</td>
</tr>
<tr>
<td>famhist</td>
<td>1</td>
<td>25.29</td>
<td>448</td>
</tr>
<tr>
<td>ns(obesity, 4)</td>
<td>4</td>
<td>3.73</td>
<td>444</td>
</tr>
<tr>
<td>ns(age, 4)</td>
<td>4</td>
<td>23.77</td>
<td>440</td>
</tr>
</tbody>
</table>
Notes

- The **degrees of freedom** fitted are the number of columns in the basis matrix (+ 1 for the intercept).
- This can also be computed as the trace of the **hat matrix**, which can be extracted from `lm`.
- This works as well for `glm`, because generalized linear models are fitted using iteratively reweighted least squares.
- §5.2.3 and §5.3 later
- fitted function \( \hat{f}_j(X_j) = h_j(X_j)^T \hat{\theta}_j \) Figure 5.4
- standard errors?
Smoothing splines (§5.4)

- ridge regression applied to natural cubic splines
- lots and lots of knots; lead to lots and lots of parameters
- regularize the solution through a penalty term

New notation: 
\[ f(X) = \sum_{j=1}^{N} \theta_j N_j(X) \] (5.10)

- knots at each distinct \( x \) value

\[
\min_{\theta} (y - N\theta)^T(y - N\theta) + \lambda \theta^T \Omega_N \theta \] (5.11)

- \( N_{ij} = N_j(x_i) \), \( \Omega_{jk} = \int N_j''(t)N_k''(t)dt \)
- note use of \( N \) for set of natural splines

- solution
\[
\hat{\theta} = (N^T N + \lambda \Omega_N)^{-1} N^T y
\]
Smoothing splines (§5.4)

- solution

$$\hat{\theta} = (N^TN + \lambda\Omega_N)^{-1}N^Ty$$

- This solves the variational problem

$$\arg\min_f \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda \int_{a}^{b} \{f''(t)\}^2 dt$$

- the solution is a natural cubic spline with knots at each $x_i$

- fitted curve

$$\hat{f}(x) = \sum_{j=1}^{N} N_j(x)\hat{\theta}_j$$

Figure 5.6
... smoothing splines

- How many parameters have been fit?
- Vector of fitted values at the training data
  \[ \hat{f} = N(N^T N + \lambda\Omega_N)^{-1} N^T y = S_\lambda y \]
- By analogy with ordinary regression, define the effective degrees of freedom (EDF) as
  \[ \text{trace } S_\lambda \]

- Smoother matrix: symmetric, positive definite
  not a projection matrix
- See p. 153–155 for more details on properties of \( S_\lambda \)
- How to choose \( \lambda \)?
  - a) Decide on degrees of freedom to be used, e.g. \texttt{smooth.spline(x, y, df=6)}, note that increasing df means less 'bias' and more 'variance'.
  - b) Automatic selection by cross-validation (Figure 5.9)
A smoothing spline version of logistic regression is outlined in §5.6, but we’ll wait till we discuss generalized additive models.

An example from the R help file for `smooth.spline`:

```r
> data(cars)
> attach(cars)
> plot(speed, dist, main = "data(cars) & smoothing splines")
> cars.spl <- smooth.spline(speed, dist)
> (cars.spl)
Call:
smooth.spline(x = speed, y = dist)

Smoothing Parameter  spar= 0.7801305  lambda= 0.1112206 (11 iterations)
Equivalent Degrees of Freedom (Df): 2.635278
Penalized Criterion: 4337.638
GCV: 244.1044
> lines(cars.spl, col = "blue")
> lines(smooth.spline(speed, dist, df=10), lty=2, col = "red")
> legend(5,120,c(paste("default [C.V.] => df =",round(cars.spl$df,1)),
+ "s( * , df = 10)"), col = c("blue","red"), lty = 1:2,
+ bg='bisque')
> detach()
```
data(cars) & smoothing splines
data(cars) & smoothing splines

![Graph showing the relationship between speed and distance with a smoothing spline line.](image-url)
data(cars) & smoothing splines

speed
dist
data(cars) & smoothing splines

- default [C.V.] => df = 2.6
- s( *, df = 10)
Multidimensional splines (§5.7)

- so far we are considering just 1 $X$ at a time
- for regression splines we replace each $X$ by the new columns of the basis matrix
- for smoothing splines we get a univariate regression
- it is possible to construct smoothing splines for two or more inputs simultaneously, but computational difficulty increases rapidly
- these are called thin plate splines
- alternative:
  \[
  E(Y \mid X_1, \ldots, X_p) = f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p)
  \]
  additive models
- binary response:
  \[
  \text{logit}\{E(Y \mid X_1, \ldots, X_p)\} = f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p)
  \]
  generalized additive models