36-617: Applied Linear Models

Nonparametric Regression I Brian Junker 132E Baker Hall brian@stat.cmu.edu

Announcements

- Project 02 Schedule:
 - □ Mon Nov 29 (or earlier): Full IDMRAD paper first draft.
 - □ Fri Dec 3: Peer reviews due.
 - □ Fri Dec 10 (or earlier): Full IDMRAD paper final draft!
- Regular classes Nov 29 & Dec 1
- No more graded hw's or quizzes
 - There is an ungraded HW11 on nonparametric regression
- BJ's remaining office hours will be in the MSP space in FMS – <u>not</u> in his office.
 - BJ will hold office hours at 11am M and W next week as well.
 The location is negotiable Shall I keep coming to FMS?

Outline

Two meanings of "Linear" model / linear smoother

$$\Box \quad y = \beta_0 + \beta_1 x + \varepsilon, \quad \text{vs.} \quad \hat{Y} = HY$$

- $\Box df = tr(H)$
- Polynomial Regression
- Fixing Collinearity: Orthogonal basis for X
- Ridge Regression as a wiggliness/roughness penalty
 Effective df = tr(H_λ)
- Cubic Regression Splines
- Variations
 - Natural Splines
 - Specifying the number of knots instead of the locations
- Smoothing Splines

Two meanings of "Linear" models

 Informally, we say a "linear" model involves a linear relationship (plus error) between x and y:

$$y_i = \beta_o + \beta_1 x_i + \varepsilon_i \tag{1}$$

A more precise and generalizable definition is that the vector ŷ is a linear function of the vector y:

$$\hat{y} = Hy \tag{2}$$

where
$$H = X(X^T X)^{-1} X^T$$
. Recall that
 $tr(H) = tr((X^T X)(X^T X)^{-1}) = p + 1 = df$

We will consider *linear models* and *linear smoothers* that generalize "linear" in the sense of (1) but preserve "linear" in the sense of (2)!

Polynomial Regression

- We already know about polynomial regression $y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots + \beta_p x_i^p + \varepsilon_i$
- This is just multiple regression where the columns of the X matrix are powers of x: 1, x, x², ..., x^p

$$X = \begin{bmatrix} 1 & \cdots & x_1^p \\ \vdots & \ddots & \vdots \\ 1 & \cdots & x_n^p \end{bmatrix}$$

and df = tr(H) = p + 1

- Good for estimating some nonlinear functions; but
 Vif's tend to be large
 - Can be overly "wiggly" or "rough"

A Running Example...

• For most of this lecture we will try to estimate f(x) from the data $y_i = f(x_i) + \varepsilon_i$ where

□
$$f(x) = (1 + 2x) \sin\left(\frac{2\pi}{0.25 + 0.75x}\right), x \in (0,1)$$

□
$$\varepsilon_i \sim N(0, \sigma^2), i = 1 ... 100$$
 ($\sigma^2 = 1$)



See the r file for this lecture for all sorts of computational details...

Polynomial regression of order 1,4,7,10





Powers of x have a lot of collinearity

> ## lmfit fits the 10th degree polynomial

- > X <- model.matrix(lmfit)</pre>
- > tr <- function(m) sum(diag(m))
- > tr(X%*%solve(t(X)%*%X)%*%t(X))
- [1] 10.9998 ## = 11, up to floating point error

> round(summary(lmfit)\$coef,2)

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-5.93	1.90	-3.12	0.00
I(x^1)	379.17	128.10	2.96	0.00
I(x^2)	-7315.39	2696.74	-2.71	0.01
I(x^3)	63240.23	26773.03	2.36	0.02
I(x^4)	-290224.39	148518.47	-1.95	0.05
I(x^5)	771227.11	499693.74	1.54	0.13
I(x^6)	-1233102.85	1058658.55	-1.16	0.25
I(x^7)	1183276.84	1419406.84	0.83	0.41
I(x^8)	-645684.02	1167866.53	-0.55	0.58
I(x^9)	171639.32	537883.43	0.32	0.75
I(x^10)	-13430.82	106143.90	-0.13	0.90

> vif(lmfit)

I(x^1)	I(x^2)	I(x^3)	I(x^4)
1.466822e+05	7.209028e+07	6.345761e+09	1.680260e+11
I(x^5)	I(x^6)	I(x^7)	I(x^8)
1.634700e+12	6.343098e+12	1.000800e+13	5.972944e+12
I(x^9)	I(x^10)		
1.130705e+12	3.958930e+10		

- The degrees of freedom are p+1 = 11, as expected
- The β̂'s and their SE's are huge
- The vif's are enormous!

Fixing collinearity: Orthogonal basis for X

- The columns of X form a basis for a p+1 dimensional subspace S of ℜⁿ: S = {v ∈ ℜⁿ st v = Xβ for some β}.
- If we replace the columns of X with orthogonal columns* that form a basis for the same space S and use them to make a model matrix Z,
 - □ Fit, prediction, etc. for the model $y = Z\beta_z + \varepsilon$ will be the same
 - The vif's should all be 1's
 - The β_z 's, $\hat{\beta}_z$'s, and $SE(\beta_z)$'s will be different, but who cares?
- The columns of Z are called "orthogonal polynomials over x"
- The R function poly(x, p) provides a set of orthogonal polynomials for us.



Polynomials tend to get too wiggly as the degree increases...



A polynomial with enough flexibility to track the more complex parts of f(x) starts to interpolate f(x) + ε, and becomes too wiggly for the less complex parts of f(x).

Digression: Review of Ridge Regression

 Recall that in Ridge Regression we are minimizing the penalized RSS

 $\sum_{i=1}^{n} (y_i - X_i \beta)^2 + \lambda \sum_{i=1}^{n} \beta^2 = (Y - X\beta)^T (Y - X\beta) + \lambda \beta^T \beta$

• Setting the derivative w.r.t. β equal to 0,

$$-2X^{T}(Y - X\beta) + 2\lambda\beta = 0$$
$$X^{T}Y = (X^{T}X + \lambda I)\beta$$
$$\widehat{\beta_{\lambda}} = (X^{T}X + \lambda I)^{-1}X^{T}Y$$

- We are effectively dividing by a function of λ , and so $\widehat{\beta_{\lambda}}$ shrinks toward zero.
- The hat matrix is $H_{\lambda} = X (X^T X + \lambda I)^{-1} X^T$, and we <u>define</u> (effective) $df = tr(H_{\lambda})$

Try to use ridge regression to control wiggliness/roughness of polynomial...



- λ = 0 corresponds to the least-squares fit (no shrinkage)
- For small positive λ, we reduce the scale of the wiggliness at the expense of some bias
- For larger values of λ, the curve is clearly shrinking towards a constant (the intercept, ≈ 0.70)
- Note how the (effective) df decreases as λ increases.

We *increase* df by increasing the polynomial degree; we *decrease* df by increasing λ .

A more "local" approach...

- Instead of fitting one curve to all the data, fit different curves to different sections of the data
 - Piecewise constant not very satisfying...
 - Piecewise polynomial still suffers from discontinuities...
 - □ →Impose continuity and smoothness constraints



We increase df either by increasing the number of intervals, or by increasing the polynomial degree

Cubic Regression Splines*

Divide x up into m+1 intervals

 $(t_0 = -\infty, t_1], (t_1, t_2], \dots, (t_{m-1}, t_m], (t_m, t_{m+1} = \infty)$ according to the <u>knots</u> t_1, t_2, \dots, t_m and consider the regression

$$y = \sum_{k=1}^{m+1} 1_{\{x \in (t_{k-1}, t_k]\}} (a_k + b_k x + c_k x^2 + d_k x^3) + \varepsilon$$
$$= \sum_{k=1}^{m+1} 1_{\{x \in (t_{k-1}, t_k]\}} p_k(x) + \varepsilon$$

subject to the constraints

□
$$p_k(t_k) = p_{k+1}(t_k)$$
 for all $k = 1 ... m$
□ $p'_k(t_k) = p'_{k+1}(t_k)$ for all $k = 1 ... m$
□ $p''_k(t_k) = p''_{k+1}(t_k)$ for all $k = 1 ... m$

*Cubic $p_k(x)$ are most popular, but obviously we could do something similar polynomials $p_k(x)$ of any degree

What are the degrees of freedom?

- Let $\beta = (a_1, b_1, c_1, d_1, \dots, d_{m+1})^T$ and let X be the matrix of functions of x, so that $y = X\beta + \varepsilon$
- There are m + 1 p_k(x)'s with 4 parameters each
 +4(m+1) columns in the X matrix
- There are 3 linear constraints on the parameters at each of the *m* knots.
 - \Box -3*m* linear constraints on the columns of X
- So we can replace the columns of X with a basis for the same subspace with only

$$4(m+1) - 3m = m+4$$

columns: the df for the spline on m knots is m+4.

What should the columns of the reduced X matrix be?

• After a little calculus^{*}, one can show that the m + 4 columns of the reduced X can be written as $1, x, x^2, x^3, (x - t_1)^3_+, \dots, (x - t_m)^3_+$, where

$$(x-t)_{+} = \begin{cases} x-t, x \ge t \\ 0, x < t \end{cases}$$

 This "basis" for X suffers from collinearity problems, and an almost-orthogonal basis of "B-splines" is usually used instead.

Our running example with cubic regression splines...

> knots <- seq(.1,.9,by=.2) ; m <- length(knots)</pre>

> pos.part <- function(x) (x+abs(x))/2</pre>

> B <- data.frame(x=x,x2=x^2,x3=x^3) ## R supplies the intercept... > for (k in knots) { B <- cbind(B,pos.part(x-k)^3) } ; B <- as.matrix(B)</pre>

- > bjfit0 <- lm(y ~ B)
- > X <- model.matrix(bjfit0)</pre>
- > df <- round(sum(diag(X%*%solve(t(X)%*%X)%*%t(X))),2)</pre>
- > ## setup() on the next line plots f(x) and the data...
- > setup(paste("handmade regression spline with",
- + length(knots), "knots &",df,"df"))
- > lines(x,predict(bjfit0))
- > bsfit0 <- lm(y ~ bs(x,knots=knots))</pre>
- > X <- model.matrix(bsfit0)
- > df <- round(sum(diag(X%*%solve(t(X)%*%X)%*%t(X))),2)</pre>
- > setup(paste("R's regression spline with",m,"knots &",df,"df"))
- > lines(x,predict(bsfit0))

> round(vif(bjfit0),2)

х	x^2		x^3	(x –	k ₁) ₊ ^3
34715.01	5734082.57	6724987	5.68	382082	202.68
$(x - k_2)_{+}^{3}$	$(x - k_3)_+^3$	$(x - k_4)_{+}^{3}$	(x – k	₅) ₊ ^3	
66989.68	3094.03	136.76		3.97	

> round(vif(bsfit0),2)

bs(x, knots = knots)1 bs(x, knots = knots)2 bs(x, knots = knots)3 2.72
3.81
4.51
bs(x, knots = knots)4 bs(x, knots = knots)5 bs(x, knots = knots)6 3.64
3.59
4.10
bs(x, knots = knots)7 bs(x, knots = knots)8 2.90
1.53



Variations on regression splines...

• **Natural splines** spend 2 more df to force $p_1(x)$ and $p_{m+1}(x)$ to be linear

> knots <- seq(.1,.9,by=.2)

> m <- length(knots)

> nsfit0 <- Im(y ~ ns(x,knots=knots))</pre>

> X <- model.matrix(nsfit0)

> df <- round(sum(diag(X%*%solve(t(X)%*%X)%*%t(X))),2)</pre>

> setup(paste("R's natural spline with",m,"knots &",df,"df"))

> lines(x,predict(bsfit0))

- If you specify df instead of knots, you get ≈ df equally spaced knots.
 - > df.req <- 9 ## select knots at (df.req 3) quantiles of x</p>
 - > bsfit1 <- lm(y ~ bs(x,df=df.req))</pre>
 - > X <- model.matrix(bsfit1)
 - > df <- round(sum(diag(X%*%solve(t(X)%*%X)%*%t(X))),2)</pre>
 - > setup(paste("R's regression spline with",df.req,"knots requested &",df,"df"))
 - > lines(x,predict(bsfit1))
 - > ## plot not shown; similar to others...





Aside: Residual diagnostics, F-test, LRT, AIC, BIC still work...

> par(mfrow=c(2,2))

> plot(nsfit0)

> plot(bsfit0)

> anova(nsfit0, bsfit0)

Analysis of Variance Table

Model 1: y ~ ns(x, knots = knots) Model 2: y ~ bs(x, knots = knots) Res.Df RSS Df Sum of Sq F Pr(>F) 1 93 115.16 2 91 108.05 2 7.1084 2.9933 0.05508 .



Be careful with F-test and LRT that the models are really nested!

Fitted values

Leverage

Smoothing Splines

 A slightly different approach to splines is to try to find a smooth function g(x) that minimizes the "Ridge-like" penalized RSS

$$\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$
 (*)

- Here, λ penalizes wiggliness or roughness: the larger λ is, the more linear g(x) must be.
- It turns out that

the function g(x) that minimizes (*) is a natural cubic spline, with knots $t_i = x_i$ at every data point x_i , and coefficients shrunken toward a linear form for g(x).

(how much shrinkage depends on λ).

Smoothing Splines and df...

- When g(x) is a natural cubic spline with knots at $t_1, ..., t_m$, the penalized RSS (*) turns out to be $(Y G\beta)^T (Y G\beta) + \lambda \beta^T M\beta$
- *G* is the X-matrix from a natural spline basis $g_1(x), ..., g_{m+2}(x)$, and *M* is a matrix with entries $M_{ij} = \int g''_i(t)g''_j(t)dt$
- Following the same calculus as for Ridge Regression, $\widehat{Y} = H_{\lambda}Y, where$ $H_{\lambda} = G \left(G^T G + \lambda M\right)^{-1} G^T$ *(Effective)* $df = tr(H_{\lambda})$

We *increase* df by increasing the sample size $x_1, ..., x_n$; we *decrease* df by increasing λ , or by using fewer knots $t_1, ..., t_m$ (at possibly different locations than the x_i 's).

Our running example: smoothing splines

> par(mfcol=c(3,2))

> setup(expression(paste("smooth spline: maximum knots, LOOCV ",

+ lambda)))

- > lines(ss <- smooth.spline(x,y,all.knots=T))</pre>
- > ss\$df ## [1] 14.76477
- > setup(expression(paste("natural spline: maximum knots, ",

+ lambda==0)))

> lines(ss <- smooth.spline(x,y,lambda=0))</pre>

```
> ss$df ## [1] 64
```

- > setup(expression(paste("smooth spline: maximum knots, ", + lambda==100)))
- > lines(ss <- smooth.spline(x,y,lambda=100))</pre>

> ss\$df ## [1] 2.002092

- > setup(expression(paste("smooth spline: ",lambda,
- + " chosen s.t. ",tr(H)==12)))

```
> lines(ss <- smooth.spline(x,y,df=12))</pre>
```

```
> ss$df ## [1] 11.99843
```

> setup(expression(paste("smooth spline: 5 knots, LOOCV ", + lambda)))

```
> lines(ss <- smooth.spline(x,y,nknots=5))</pre>
```

> ss\$df ## [1] 5.784098

- > setup(expression(paste("natural spline: 5 knots, ", + lambda==0)))
- > lines(ss <- smooth.spline(x,y,nknots=5,lambda=0))</pre>

> ss\$df ## [1] 7







By default, smooth.spline() chooses as many equally-spaced knots as it can, up to the sample size n. If you set all.knots=TRUE, smooth.spline() uses all the data points as knots.

smooth.spline() is part of the base R distribution.

Summary

Two meanings of "Linear" model / linear smoother

$$\Box \quad y = \beta_0 + \beta_1 x + \varepsilon, \quad \text{vs.} \quad \hat{Y} = HY$$

- $\Box df = tr(H)$
- Polynomial Regression
- Fixing Collinearity: Orthogonal basis for X
- Ridge Regression as a wiggliness/roughness penalty
 Effective df = tr(H_λ)
- Cubic Regression Splines
- Variations
 - Natural Splines
 - Specifying the number of knots instead of the locations
- Smoothing Splines