STAT 705 Nonlinear regression

Adapted from Timothy Hanson

Department of Statistics, University of South Carolina

Stat 705: Data Analysis II

Chapter 13 Parametric nonlinear regression

Throughout most of STAT 704 and 705, we concentrated on linear models where $E(Y_i) = \mathbf{x}_i'\boldsymbol{\beta}$. Notable exceptions arose when we considered non-normal data. For logistic regression we had $E(Y_i) = e^{\mathbf{x}_i'\boldsymbol{\beta}}/[1 + e^{\mathbf{x}_i'\boldsymbol{\beta}}]$; Poisson regression gave us $E(Y_i) = t_i e^{\mathbf{x}_i'\boldsymbol{\beta}}$.

Sometimes scientists have a parametric non-linear mean function in mind for normal data. Theoretical considerations may lead to such a model, or else empirical evidence collected over time. Examples: dose-response models, growth curves, heating in swine due to MRI.

Parametric nonlinear regression

A parametric nonlinear model (13.1–13.5) has a prespecified parametric form indexed by parameters $\boldsymbol{\gamma}$

$$Y_i = f(\mathbf{x}_i, \boldsymbol{\gamma}) + \epsilon_i.$$

For example the exponential growth/decay model is $Y_i = \gamma_0 e^{\gamma_1 x_i} + \epsilon_i$. Data reduction takes place through the estimation of $\boldsymbol{\gamma} = (\gamma_0, \gamma_1)$ and σ.

Other examples are the logistic growth curve $Y_i = \gamma_0[1 + \gamma_1 \exp(\gamma_2 x_i)]^{-1} + \epsilon_i$ and the von Bertlanffy growth curve $Y_i = L_\infty [1 - \exp(-K(x_i - x_0))] + \epsilon_i$.

Note that model diagnostics are similar to the linear case, for example $r_i = Y_i - f(\mathbf{x}_i, \hat{\boldsymbol{\gamma}})$ can be used to assess model adequacy.

Fitting parametric nonlinear models

Fitting of such models is carried out via maximum likelihood using Newton-Raphson. Several functions in SAS can carry this out; PROC NLMIXED is the most versatile, while PROC NLIN is the old-school workhorse. Good starting values can make or break the program (as we'll see); you need to think about what the parameters represent in the model.

There is a bit on fitting at the end of the logistic regression notes. In your book see pp. 517–521. This theory is covered in more detail in STAT 823 (large sample theory) and STAT 740 (advanced statistical computing).

PROC NLMIXED provides the MLE's as well as standard errors. Also, functions of parameters can be estimated as well.

Example. Yellowfin Tuna growth curve data

Yellowfin tuna are an important commercial and recreational fishery; regulation of tuna fisheries worldwide is managed by ICCAT (International Commission on the Conservation of Atlantic Tuna)

Example. Yellowfin Tuna growth curve data

Proper management can include size limits on catch to, e.g., ensure fish have had at least one chance to reproduce before capture. Growth curve models relating length (easily measured) to age (very difficult to measure) are valuable for fisheries management decisions.

The von Bertalanffy growth curve is popular in fisheries management (with x_i typically replaced by t). Parameters are L_∞, the growth limit; K, the growth rate; and x_0, the age at length 0. Reasonable start values for L_∞ and x_0 are straightforward; with a little math, for some reasonable x_i, try $k_{init} = log \frac{Y_{x_i} - L_\infty}{Y_{x_i+1} - L_\infty}$

Example with SAS code

The yellowfin tuna data come from two separate populations in the Atlantic and Pacific oceans. We have more data for the Pacific population.

```
data yellowfin;
input Age Length Ocean $ @@
datalines;
 803 179 Atlantic 789 182 Atlantic 644 173 Atlantic
...
1354 251  Pacific 221 160  Pacific 202 145  Pacific
;

proc sgplot data=yellowfin;
scatter x=age y=length/group=ocean;
xaxis min=0; yaxis min=0; run;

*NLMIXED did not converge at first--used NLIN instead;
proc nlin data=yellowfin plots=all; where ocean='Pacific';
parms Linf=200 to 300 by 50 x0=-20 to 20 by 10 K=0.001 to 0.004 by 0.001;
expo=exp(-K*(age-x0));
mu=Linf*(1-expo);
model length = mu;
der.Linf=1-expo;
der.K=Linf*(age-x0)*expo;
der.t0=-Linf*K*expo;
output out=fit pred=Lpred;
run;
```

Problem from 2010 MS/PhD qualifying exam

The May 2010 qualifying exam (part II) has a nice problem.

```
data snake;
input conc rate @@;
datalines;
 31.25 53.01 62.5 81.42 125 122.11 250 304.57 500 376.87
 1000 414.13 2000 553.46
;

proc sgscatter; plot rate*conc;

proc nlmixed data=snake;
 parms b1= b2= b3= sigma=; * let's figure these out in CE 10;
 mu=b1/(1+(b2/conc)**b3);
 model rate ~ normal(mu,sigma*sigma);
 predict b1/(1+(b2/conc)**b3) out=fit;
 estimate "mean rate at conc=750" b1/(1+(b2/750)**b3);

proc sgplot data=fit;
 scatter x=conc y=rate;
 series x=conc y=pred;
```

Nonparametric regression

Consider a continuous response with three predictors (although these methods can be extended to other types of response).

An additive model stipulates

$$Y_i = \mu + f_1(x_{i1}) + f_2(x_{i2}) + f_3(x_{i3}) + \epsilon_i,$$

and seeks to estimate the functions $f_1(x)$, $f_2(x)$, and $f_3(x)$ (typically via splines). These are fit in `proc gam` and `proc transreg`. We can also consider a transformation of Y_i as well as pairwise interaction surfaces.

Nonparametric regression

A parametric nonlinear model (Chapter 13) has a prespecified parametric form indexed by parameters $\boldsymbol{\gamma}$

$$Y_i = f(\mathbf{x}_i, \boldsymbol{\gamma}) + \epsilon_i.$$

For example the exponential growth/decay model is $Y_i = \gamma_0 e^{\gamma_1 x_i} + \epsilon_i$. Data reduction takes place through the estimation of $\boldsymbol{\gamma}$ and σ.

Nonparametric regression is essentially unspecified

$$Y_i = f(\mathbf{x}_i) + \epsilon_i,$$

and seeks to estimate $f(\mathbf{x}) : \mathbb{R}^k \to \mathbb{R}$ directly. Two useful and popular methods are *lowess* and *kernel smoothing*.

Kernel smoothing

Let's start with a univariate predictor yielding data $\{(x_i, Y_i)\}_{i=1}^n$. At each $x \in \mathbb{R}$, the kernel-smoothed estimate of $f(\cdot)$ is a weighted average of the Y_i's:

$$\hat{f}_h(x) = \sum_{i=1}^{n} \left[\frac{k\{(x_i - x)/h\}/h}{\sum_{j=1}^{n} k\{(x_j - x)/h\}/h} \right] Y_i.$$

Here, $k(d)$ is the kernel. Common choices are Gaussian $k(d) = e^{-0.5d^2}$ (most common), uniform $k(d) = I\{|d| < 1\}$, and Epanechnikov $k(d) = 0.75(1 - d^2)I\{|d| < 1\}$ (there are many more). Different kernel functions simply weight neighboring points differently.

Bandwidth

The parameter h is called the bandwidth. The larger the bandwidth, the smoother the estimate $\hat{f}_h$. What happens to $\hat{f}_h$ as $h \to \infty$? Is it possible to have $\hat{f}_h(x)$ outside the range of Y_i-values?

A common way to choose the bandwidth is through cross-validation, $\hat{h} = \text{argmin}_{h>0} \sum_{i=1}^{n} (Y_i - \hat{f}_{h,i}(x_i))^2$ where $\hat{f}_{h,i}$ is the kernel-smoothed estimate based on the $(n-1)$ pairs $\{(x_j, Y_j)\}_{j \neq i}$.

`ksmooth` in R gives kernel-smoothed regression estimates without standard errors. A great package that does a lot more (including handling categorical predictors) is `np`. You need to install it from CRAN.

Yellowfin tuna example in R with Gaussian kernel-smoothing

Recall that Y_i is length and x_i is age. The default bandwidth h selection is cross-validation; surprisingly, I found it under-smoothed the data. The default kernel is Gaussian.

```
library(np)
Tuna.df <- read.delim(‘‘Yellowfin.txt’’,header=T)
attach(Tuna.df)
Length_Pacific <- Length[Ocean==’’Pacific’’]
Age_Pacific <- Age[Ocean==’’Pacific’’]
fit1 <-npreg(Length_Pacific~Age_Pacific)
plot(fit1,plot.errors.method="asymptotic",plot.errors.style="band",main="Kernel-smoothed")
points(Age_Pacific,Length_Pacific)
```

11.4 **LO**cally **WE**ighted **S**catterplot **S**moothing (lowess)

Kernel-smoothing is biased at the boundaries $\min\{x_i\}$ and $\max\{x_i\}$, and at the extrema of $f(\cdot)$. A method that solves some of these issues uses locally fitted polynomials to estimate $f(x)$ at each x via weighted least squares (WLS). Lowess was introduced by Cleveland (1979).

Recall that weighted least squares weights some pairs (x_i, Y_i) more heavily when "more information" is known about Y_i, e.g. $var(Y_i)$ is smaller than for other values. The weight w_i attached to (x_i, Y_i) is the ith diagonal of the matrix $\mathbf{W}$; the remaining elements are zero. The weighted least squares estimate of β is given by $\hat{\beta} = (\mathbf{XWX}')^{-1}\mathbf{X}'\mathbf{WY}$.

lowess

Consider estimating $f(x)$ at x with a linear or quadratic function. If we assume that pairs (x_i, Y_i) have more information for $f(x)$ at values of x_i near x, we can weight them more using WLS. The most common weight function is tricube

$$w_i(x) = \left\{ \begin{array}{cc} [1 - (|x - x_i|/d_q(x))^3]^3 & |x - x_i| < d_q(x) \\ 0 & |x - x_i| > d_q(x) \end{array} \right\}.$$

$d_q(x)$ is a distance such that the proportion of x_i values within x is q, i.e. $d_q(x) = \min\{d > 0 : \frac{1}{n}\sum_{i=1}^{n} I\{|x_i - x| < d\} \geq q\}$. A common choice of q is 0.5 (p. 450).

The lowess estimate of $f(x)$, assuming local linear fitting, is then $\hat{f}(x) = [\ 1 \quad x\ ](\mathbf{X}\mathbf{W}(x)\mathbf{X}')^{-1}\mathbf{X}'\mathbf{W}(x)\mathbf{Y}$ where $\mathbf{W}(x) = \text{diag}(w_1(x), \ldots, w_n(x))$ and the ith row of bX is $[\ 1 \quad x_i\ ]$. For *each value* of x, a separate WLS is fitted – lowess requires *a lot* of computation!

Yellowfin tuna example in R with lowess

This uses defaults, which actually over-smooth in this case (`enp.target` can be manipulated to fix this). An older function is `lowess`; `loess` has improvements on `lowess` but gives essentially the same answers.

```
fit2=loess(Length_Pacific~Age_Pacific)
pred.Age=seq(0,1200,20)
pred2=predict(fit2,pred.Age,se=TRUE)
plot(pred.Age,pred2$fit,type="l",xlab="Age",ylab="Length",main="Lowess Fit")
lines(pred.Age,pred2$fit-1.96*pred2$se.fit,lty=3)
lines(pred.Age,pred2$fit+1.96*pred2$se.fit,lty=3)
points(Age_Pacific,Length_Pacific)
```

Similarities between lowess and kernel-smoothing

Both kernel-smoothing and lowess have weight functions and bandwidths that determine how points in a neighborhood of x are weighted.

Both estimates are written as $\hat{f}(x) = \mathbf{c}(x)'\mathbf{Y}$, i.e. are linear combinations of the Y_i's that depend on $\mathbf{x}$. In STAT 704 regression, $\hat{f}(x) = \mathbf{c}(x)'\mathbf{Y}$ where $\mathbf{c}(x)' = [\ 1 \quad x\ ](\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$. Note that kernel-smoothing provides a true average of the Y_i's at each point, whereas lowess values of $c_i(x)$ may be negative or greater than one.

Both methods are generalized to more than one predictor similarly. Predictors are standardized to have variance one and Euclidean distance $d = ||\mathbf{x} - \mathbf{x}^*||$ is used in the weight function rather than $|x - x^*|$, or else the Mahalanobis distance is used $d = \sqrt{(\mathbf{x} - \mathbf{x}^*)'\mathbf{S}^{-1}(\mathbf{x} - \mathbf{x}^*)}$ (no need to standardized first). Note that categorical predictors need some thought.

Questions and comments

- Is extrapolation a good idea with lowess or kernel-smoothed methods?
- The asymptotics for nonparametric smoothing methods is worth an entire course. A bit is covered in STAT 824 (nonparametrics).
- Which method, lowess or kernel-smoothing, is more appropriate for Bernoulli data? Why?
- There's some nice animation here: http://www.r-bloggers.com/some-heuristics-about-local-regression-and-kernel-smoothing/
- A method worthy of its own lecture is *basis expansions*. Basis expansions write the unknown $f(\cdot)$ as $f(\mathbf{x}) = \sum_{k=1}^{K} \beta_k \phi_k(\mathbf{x})$ for a set of known functions $\phi_k(\cdot)$. The unknown parameters are $\beta_1, \ldots, \beta_K$. *This yields a linear model*.
- Example basis expansions include polynomials, Legendre polynomials, wavelets, sines and cosines, and B-splines.