Nonlinear Statistical Models

Earlier in the course, we considered the general linear statistical model (GLM):

\[ y_i = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_k x_{ki} + \epsilon_i, \quad i = 1, \ldots, n, \]

written in matrix form as:

\[
\begin{bmatrix}
  y_1 \\
  \vdots \\
  y_n
\end{bmatrix} =
\begin{bmatrix}
  1 & x_{11} & \cdots & x_{1p-1,1} \\
  1 & x_{12} & \cdots & x_{1p-1,2} \\
  \vdots & \vdots & \ddots & \vdots \\
  1 & x_{1n} & \cdots & x_{1n,p-1,1}
\end{bmatrix}
\begin{bmatrix}
  \beta_0 \\
  \beta_1 \\
  \vdots \\
  \beta_{p-1}
\end{bmatrix}
+ \begin{bmatrix}
  \epsilon_1 \\
  \epsilon_2 \\
  \vdots \\
  \epsilon_n
\end{bmatrix}
\quad \text{or} \quad y = X\beta + \epsilon.
\]

Specifically, we discussed estimation of the model parameters, measurements of uncertainty in these estimates and the predicted values, and the use of confidence intervals or significance tests to assess the significance of parameters. Through the matrix formulation, we essentially saw that in linear models (where the model is a linear function of the parameters), the estimates of linear functions of the response values, and the standard errors of these estimates have simple closed-form expressions and can thus be analytically computed. Unfortunately, this will not be the case for nonlinear models.

Definition: As before, consider a set of \( n \) observed values on a response variable \( y \) assumed to be dependent on \( k \) explanatory variables \( x = (x_1, \ldots, x_k) \). This situation can be described through the following nonlinear statistical model:

\[ y_i = f(x'_i, \theta) + \epsilon_i, \quad i = 1, \ldots, n, \]

where \( x'_i = [x_{1i}, x_{2i}, \ldots, x_{ki}] \) is the row vector of observations on \( k \) explanatory variables \( x_1, \ldots, x_k \) for the \( i^{th} \) observational unit, \( \theta = (\theta_1, \ldots, \theta_p) \) is a vector of \( p \) parameters, and \( \epsilon_i \) is the \( i^{th} \) error term. In a matrix setting, this model can be expressed as:

- Typically, we assume \( \epsilon_i \sim (0, \sigma^2) \) and that the explanatory variables in \( x \) are fixed.
- If \( f(\theta) = X\theta \), where \( X \) is the design matrix as defined earlier, then the model reverts to a linear model as before.
- As Leonid has argued, nonlinear models commonly arise as solutions to a system of differential equations which describe some dynamic process in a particular application. Some common examples of nonlinear models are given below:

1. **Exponential Growth/Decay Model**: \( y_i = \alpha e^{\beta x_i} + \epsilon_i \), where \( x \) is typically time.
   - This arises as a solution of the differential equation \( \frac{\partial E(y)}{\partial x} = \beta E(y) \).

2. **Two-term Exponential Model**: \( y_i = \frac{\theta_1}{\theta_1 - \theta_2} \left[ e^{-\theta_2 x_i} - e^{-\theta_1 x_i} \right] + \epsilon_i \).
This model results when two processes lead to exponential growth or decay. For example, if we are monitoring the concentration of a drug in the bloodstream, which enters the bloodstream from muscle tissue and is removed by the kidneys, then this model is a solution to the differential equation:

\[
\frac{\partial E(y)}{\partial x} = \theta_1 E(y_m) - \theta_2 E(y),
\]

where \( y_m \) is the concentration of the drug in the muscle tissue and \( y \) is the concentration in the blood.

3. Mitscherlich Model: \( y_i = \alpha \left[ 1 - e^{-\beta(x_i+\delta)} \right] + \epsilon_i \).

As an example of where this model is used, let \( y \) be the yield of a crop, let \( x \) be the amount of some nutrient, and let \( \alpha \) be the maximum attainable yield. Then if we assume the increase in yield is proportional to the difference between \( \alpha \) and the actual yield, we get the following differential equation, to which the model is a solution:

\[
\frac{\partial E(y)}{\partial x} = \beta [\alpha - E(y)],
\]

where \( \delta \) is the equivalent nutrient value of the soil.

4. Logistic Growth Model: \( y_i = \frac{Mu}{u + (M - u) \exp(-kt_i)} + \epsilon_i \).

This model results from a differential equation where we assume the population growth rate decays as the population increases, so that it reaches a carrying capacity \( M \). The form of this differential equation was discussed in some detail in the past weeks, and is given by:

\[
\frac{dE(y)}{dt} = ku \left( 1 - \frac{u}{M} \right),
\]

where \( M \) is the carrying capacity, \( u \) is the population size at time 0, and \( k \) controls the rate of growth. The form of this model, is shown as a fit to the AIDS data (1981-1992) below.

![AIDS Cases: 1981–1992](image_url)
In principle, fitting a nonlinear statistical model using least squares is no different than fitting a linear one. That is, for the model \( y_i = f(x_i', \theta) + \epsilon_i \), we seek to find that parameter vector \( \hat{\theta} \) that minimizes the sum of the squared errors:

\[
Q(\theta | y, X) = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} [y_i - f(x_i', \theta)]^2 = [y - f(\theta)]'[y - f(\theta)] = \epsilon'\epsilon
\]

where \( f(\theta) \) is the \( n \times 1 \) vector of \( f(x_i', \theta) \) values. Differentiating \( Q \) with respect to the \( \theta_j \) (in an effort to minimize \( Q \), setting the resulting equations to zero and evaluating at \( \theta = \hat{\theta} \) gives:

\[
\frac{\partial Q(\theta | y, X)}{\partial \theta_j} \bigg|_{\theta = \hat{\theta}} = -2 \sum_{i=1}^{n} [y_i - f(x_i', \hat{\theta})] \left[ \frac{\partial f(x_i', \hat{\theta})}{\partial \theta_j} \right] = 0, \forall j = 1, \ldots, p.
\]

These resulting equations are \( p \) nonlinear equations in the \( p \) parameters, which in general cannot be solved analytically to obtain explicit solutions for \( \theta \). As a result, numerical methods are used to solve these nonlinear systems. In MatLab, there are several numerical methods that can be used, although the default is a generalization of the standard Gauss-Newton method, known as the Levenberg-Marquardt Algorithm.

**Gauss-Newton Method:** This method, like others that may be used, is iterative in nature, and requires starting values for the parameters. These starting values are then continually adjusted to home in on the least squares solution. The method is conducted using the following steps:

1. Consider taking a Taylor expansion of \( f(\theta) \) about some starting parameter vector \( \theta^0 \):

\[
f(\theta) \approx f(\theta^0) + F(\theta^0)(\theta - \theta^0), \tag{1}
\]

where \( F(\theta^0) \) is the \( n \times p \) matrix of partial derivatives evaluated at \( \theta^0 \) and the \( n \) data values \( x_i' \), as given below:

\[
F(\theta^0) = \begin{bmatrix}
\frac{\partial f(x_1', \theta^0)}{\partial \theta_1} & \frac{\partial f(x_1', \theta^0)}{\partial \theta_2} & \cdots & \frac{\partial f(x_1', \theta^0)}{\partial \theta_p} \\
\frac{\partial f(x_2', \theta^0)}{\partial \theta_1} & \frac{\partial f(x_2', \theta^0)}{\partial \theta_2} & \cdots & \frac{\partial f(x_2', \theta^0)}{\partial \theta_p} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial f(x_n', \theta^0)}{\partial \theta_1} & \frac{\partial f(x_n', \theta^0)}{\partial \theta_2} & \cdots & \frac{\partial f(x_n', \theta^0)}{\partial \theta_p}
\end{bmatrix}_{n \times p}.
\]

2. Calculate \( f(\theta^0) \) and \( F(\theta^0) \). Recognizing \( f(\theta) \) as an approximately linearized model in \( \theta \) from equation (1), linear least squares is used to update the starting values and obtain the solution at the next iteration.

3. Repeat steps 1 & 2 until the change in the parameters is below some tolerance.
Some Key Results: Based on this same Taylor expansion, Gallant (1987) established the following asymptotic (large sample) results:

1. If we assume \( \epsilon \sim N(0, \sigma^2 I) \), then the least squares parameter vector \( \hat{\theta} \) is approximately normally distributed with mean \( \theta \) and variance matrix \( \text{Var}(\hat{\theta}) = (F'F)^{-1}\sigma^2 \), written:
   \[
   \hat{\theta} \sim N(\theta, (F'F)^{-1}\sigma^2)
   \]
   - In practice, we compute \( F(\hat{\theta}) \) as an estimate of \( F(\theta) \) and estimate \( \sigma^2 \) with \( \text{MSE} = \frac{\text{SSE}}{n-p} \) to give the asymptotic variance-covariance matrix for \( \hat{\theta} \) as:
     \[
     s^2(\hat{\theta}) = \text{Var}(\hat{\theta}) = (\hat{F}'\hat{F})^{-1} \cdot \text{MSE}
     \]
   - This gives a \( 100(1-\alpha)\% \) confidence interval for a given parameter \( \theta_i \) as:
     \[
     \hat{\theta}_i \pm t_{1-\alpha,n-p} \cdot s(\hat{\theta})_{ii} = \hat{\theta}_i \pm t_{1-\alpha,n-p} \cdot \sqrt{\text{MSE}(\hat{F}'\hat{F})^{-1}}
     \]
     where \( (\hat{F}'\hat{F})^{-1} \) is the \( i^{th} \) diagonal entry of \( (\hat{F}'\hat{F})^{-1} \).

2. Again assuming the errors \( \epsilon \sim N(0, \sigma^2 I) \), then: \( \text{Var}(\bar{y}) = [F(F'F)^{-1}F']\sigma^2 \), which can be estimated as: \( s^2(\bar{y}) = \text{Var}(\bar{y}) = [\hat{F}(\hat{F}'\hat{F})^{-1}\hat{F}'] \cdot \text{MSE} \).
   - This technique of quantifying uncertainty through the use of a first-order Taylor series expansion is known as the Delta Method and was the most commonly used technique for finding approximate standard errors before the advent of modern computing.
   - Gallant (1987) illustrated several cases where these approximations can be very poor in practice, giving confidence intervals and hypothesis tests which are completely wrong.
   - Some more modern techniques for estimating standard errors and subsequently performing statistical inferences in nonlinear models are the method of bootstrapping, and Markov chain Monte Carlo (MCMC) methods.

Example: Data were collected on the number of big horn sheep on the Mount Everts winter range in Yellowstone National Park between 1965 and 1999 (obtained from Marty Kardos). Consider the time plot of these data to the right. The population suffered a die-off resulting from the infectious keratoconjunctivitis in 1983. Marty indicates that this is fairly typical for bighorn sheep populations: logistic-like population growth up to a carrying capacity followed by disease-related die-offs,
and subsequent recovery. For now, consider modeling the initial population growth from 1965 to 1982 using a logistic growth model. Here, after discussing the fit of the model, this example is used to illustrate the calculation of standard errors for the parameter estimates from this nonlinear model.

The logistic growth model is a 3-parameter nonlinear model of the form:

\[ y_i = \frac{Mu}{u + (M - u) \exp(-kt_i)} + \epsilon_i, \]

where:

- \( y_i \) = the number of bighorn sheep at time \( t_i \),
- \( M \) = the carrying capacity of the population,
- \( u \) = the number of sheep at time 0 (1964 here),
- \( k \) = a parameter representing the growth rate.

Fitting this model to the 1965-1982 data with nonlinear least squares using the `nlinfit` function in MatLab gives the following fitted model:

\[ \hat{y}_i = \frac{\hat{M}\hat{u}}{\hat{u} + (\hat{M} - \hat{u}) \exp(-\hat{k}t_i)}, \]

where \( \hat{M} = 221.0513 \), \( \hat{u} = 27.4643 \), \( \hat{k} = 0.3279 \), as computed using the Gauss-Newton method outlined earlier. The fitted model is shown in overlay to the right. Is this a good model?

The code for fitting the logistic model is given below and can be found on the course webpage under the files `sheep.m` and `logistic.m`. The first file is the driver program and the `logistic.m` file defines the logistic function.

```matlab
% Fits a logistic growth model to the 1965-1982 sheep data, and plots the fitted model in overlay.

sheep82.year = sheep.year(sheep.year<=1982); % Take only years before 1983
sheep82.count = sheep.count(sheep.year<=1982); % Take only cases before 1983
time = sheep82.year-1964; % Defines the year-1964 variable
beta = [221 27 0.33]; % Parameter starting values
[betahat,resid,J] = nlinfit(time, ... % Performs NLS, returning the parameter estimates, residuals, & Jacobian
sheep82.count,@logistic,beta); %
```

61
% Computes predicted counts
plot(sheep82.year,sheep82.count,'ko',... % Plots the sheep counts vs. year
sheep82.year,yhat); % with fitted curve in overlay
xlabel('Year','fontsize',14); % Puts an x-axis label on plot
ylabel('Number of Bighorn Sheep','fontsize',14);% Puts a y-axis label on plot
title('Bighorn Sheep Counts: 1965-1982',... % Puts a title on the plot
'fontsize',14,'fontweight','b');

- I tried a few different things to try to improve the fit, including a “Weibull-like” modification to the logistic growth model, where I added a parameter in the exponent of the model yielding a **generalized logistic growth model**:

\[
y_i = \frac{Mu}{u + (M - u) \exp[-kt_i^\gamma]} + \epsilon_i,
\]

where \( \gamma \) is a parameter controlling the degree of curvature in the logistic shape. If \( \gamma = 1 \), this reverts to the usual logistic growth model.

- This change is “Weibull-like” in the sense that we have added a parameter in the exponent of an exponential function, much like the addition of the \( \gamma \)-parameter in going from exponential growth to a Weibull growth model.

- This “Weibull-like” curvature-adjustment to the logistic model has a corresponding differential equation to which it is the solution. This equation has the form:

\[
\frac{dE(y)}{dt} = \gamma t^{\gamma-1}ku \left(1 - \frac{u}{M}\right),
\]

where \( M \) is the carrying capacity, \( u \) is the population size at time 0, \( k \) controls the rate of growth, and \( \gamma \) is this curvature piece.

- When parameterized this way, it turns out that the resulting parameter estimates for \( k \) & \( \gamma \) are unstable because they are highly correlated. An alternative, but equivalent way to parameterize this model is as follows:

\[
y_i = \frac{Mu}{u + (M - u) \exp[-(kt_i)^\gamma]} + \epsilon_i.
\]

This is the form of the model we will use for the remainder of our treatment of this generalized logistic model.

- Essentially, \( \gamma \) controls the **steepness** of the S-shape in the logistic model, where larger values of \( \gamma \) correspond to steeper curves. Since our logistic model for the sheep data appears to not be steep enough, we would expect this parameter to be larger than 1.

- Fitting this generalized logistic model produced the fitted curve and parameter estimates as given on the next page.
Does this seem like a better fit?

The MatLab code used to fit this generalized logistic model is shown below:

```matlab
beta2 = [221 27 0.3 2];
[betahat2,resid2,J2] = nlinfit(time, sheep82.count,@genlogistic,beta2);
yhat2 = sheep82.count-resid2;
plot(sheep82.year,sheep82.count,'ko',sheep82.year,yhat2);
xlabel('Year','fontsize',14);
ylabel('Number of Bighorn Sheep','fontsize',14);
title('Bighorn Sheep Counts (1965-1982) with Generalized Logistic Fit','fontsize',14,'fontweight','b');
```

Which of these parameters do you expect to be the least stable (i.e.: which do you expect to have the highest variability?) To get the standard errors of these parameters in \( \hat{\theta} = (M, u, k, \gamma) \), we need to compute the Delta Method approximate variance matrix, given as:

\[
s^2(\hat{\theta}) = \text{Var}(\hat{\theta}) = (\hat{F}'\hat{F})^{-1} \cdot \text{MSE}
\]

To compute the \( \hat{F} \)-matrix, we first compute the partial derivatives of

\[
f_i(\theta) = f(x'_i, \theta) = \frac{Mu}{u + (M - u) \exp\left[-(kt_i)^\gamma\right]}
\]

with respect to \( M, u, k, \) and \( \gamma \) respectively. Doing so with repeated use of the product (quotient) rule gives the following:

\[
\begin{align*}
\frac{\partial f_i(\theta)}{\partial M} &= \frac{u^2\{1-\exp\left[-(kt_i)^\gamma\right]\}}{\{u + (M - u) \exp\left[-(kt_i)^\gamma\right]\}^2}, \\
\frac{\partial f_i(\theta)}{\partial u} &= \frac{M^2 \exp\left[-(kt_i)^\gamma\right]}{\{u + (M - u) \exp\left[-(kt_i)^\gamma\right]\}^2}, \\
\frac{\partial f_i(\theta)}{\partial k} &= \frac{Mu(M-u)\gamma k^{-1}t_i^\gamma \exp\left[-(kt_i)^\gamma\right] }{\{u + (M - u) \exp\left[-(kt_i)^\gamma\right]\}^2}, \\
\frac{\partial f_i(\theta)}{\partial \gamma} &= \frac{Mu(M-u)(kt_i)^\gamma \exp\left[-(kt_i)^\gamma\right] \log(kt_i)}{\{u + (M - u) \exp\left[-(kt_i)^\gamma\right]\}^2}.
\end{align*}
\]
Plugging the least squares estimates $\hat{\theta} = (\hat{M}, \hat{u}, \hat{k}, \hat{\gamma})$ into these derivatives yields the $F$-matrix shown below:

$$F = \begin{bmatrix}
\frac{\partial f(t_1, \hat{\theta})}{\partial \hat{M}} & \frac{\partial f(t_1, \hat{\theta})}{\partial \hat{u}} & \frac{\partial f(t_1, \hat{\theta})}{\partial \hat{k}} & \frac{\partial f(t_1, \hat{\theta})}{\partial \hat{\gamma}} \\
\frac{\partial f(t_2, \hat{\theta})}{\partial \hat{M}} & \frac{\partial f(t_2, \hat{\theta})}{\partial \hat{u}} & \frac{\partial f(t_2, \hat{\theta})}{\partial \hat{k}} & \frac{\partial f(t_2, \hat{\theta})}{\partial \hat{\gamma}} \\
\vdots & \vdots & \vdots & \vdots \\
\frac{\partial f(t_{17}, \hat{\theta})}{\partial \hat{M}} & \frac{\partial f(t_{17}, \hat{\theta})}{\partial \hat{u}} & \frac{\partial f(t_{17}, \hat{\theta})}{\partial \hat{k}} & \frac{\partial f(t_{17}, \hat{\theta})}{\partial \hat{\gamma}}
\end{bmatrix}
$$

where the actual values were computed in MatLab. Finally, computing $\text{MSE} \cdot (F'F)^{-1}$ to get the variance-covariance matrix of the parameter estimates, extracting the diagonal entries of this 3x3 matrix, and taking square roots yields the standard errors of the parameter estimates. The MatLab code to do this is given below:

```matlab
% Computes the Delta Method parameter estimate standard errors for a 4-parameter generalized logistic model.

% % % %
% % % %
% n = length(sheep82.year); % Sample size
% p = length(beta2); % Number of model parameters
% mse = sum(resid2.^2)/(n-p); % Computes the mean squared error (MSE)
% Mhat = betahat2(1); uhat = betahat2(2); % Renames the three parameter estimates
% khat = betahat2(3); ghat = betahat2(4); %
% den = (uhat+(Mhat-uhat)*exp(-(khat*... % Denominator of first-order time).^ghat)).^2; %
% dfdM = uhat.^2*(1-exp(-(khat*time).^... % Computes df/dM (nx1 vector)
% ghat))./den;
% dfdu = Mhat.^2*exp(-(khat*time).^... % Computes df/du (nx1 vector)
% ghat).*ghat.*khat.^(-ghat-1).*... %
% exp(-(khat*time).^ghat))./den;
% dfdk = Mhat*uhat*(Mhat-uhat).*time.^... % Computes df/dk (nx1 vector)
% ghat).*ghat.*khat.^(-ghat-1).*... %
% exp(-(khat*time).^ghat))./den;
% dfdg = Mhat*uhat*(Mhat-uhat).*time.^... % Computes df/dg (nx1 vector)
% ghat.*log(khat*time).*... %
% exp(-(khat*time).^ghat))./den;
% Fhat = [dfdM dfdu dfdk dfdg]; % Combines derivatives into nx4 matrix
% varmat = inv(Fhat'*Fhat).*mse; % Inverts estimated (F'F)*MSE matrix
% separ = sqrt(diag(varmat)); % Standard errors are square root
```

$$\begin{bmatrix}
0.0001 & 1.0004 & 0.0012 & -0.0969 \\
0.0053 & 1.0161 & 0.0533 & -1.9492 \\
0.0153 & 1.0415 & 0.1532 & -3.6359 \\
0.0371 & 1.0816 & 0.3439 & -5.0945 \\
0.0825 & 1.1242 & 0.6710 & -5.0515 \\
0.1729 & 1.1274 & 1.1459 & -1.5678 \\
0.3349 & 1.0106 & 1.6292 & 6.4654 \\
0.5652 & 0.7222 & 1.7487 & 15.1716 \\
0.7904 & 0.3683 & 1.2833 & 16.5367 \\
0.9293 & 0.1276 & 0.6181 & 10.3187 \\
0.9834 & 0.0302 & 0.1980 & 3.9937 \\
0.9973 & 0.0050 & 0.0427 & 0.9987 \\
0.9997 & 0.0006 & 0.0061 & 0.1615 \\
1.0000 & 0.0000 & 0.0000 & 0.0165 \\
1.0000 & 0.0000 & 0.0000 & 0.0010 \\
1.0000 & 0.0000 & 0.0000 & 0.0000 \\
1.0000 & 0.0000 & 0.0000 & 0.0000 \\
\end{bmatrix}$$
tstar = tinv(1-.025/p,n-p); % Computes Bonferroni t-value
ci = [betahat2'-tstar*separ betahat2'+ ... % Computes Bonferroni CIs for the
tstar*separ]; % three model parameters

betahat2 = 207.640 67.973 0.1380 3.4542
separ = 4.178 7.166 0.0091 0.8093

• This provides approximate Bonferroni-adjusted *simultaneous* 95% confidence intervals for the four parameters (using $t_1-.025/4(n - p) = t_{.99375}(17 - 4) = t_{inv}.99375,13 = 2.8961$) as:

\[
\begin{align*}
\hat{M} & \pm t_{.99375}(n - p) \cdot \text{SE}(\hat{M}) = 207.640 \pm 2.8961(4.178) = (195.54, 219.74). \\
\hat{u} & \pm t_{.99375}(n - p) \cdot \text{SE}(\hat{u}) = 67.973 \pm 2.8961(7.166) = (47.22, 88.73). \\
\hat{k} & \pm t_{.99375}(n - p) \cdot \text{SE}(\hat{k}) = 0.1380 \pm 2.8961(0.0091) = (0.1116, 0.1645). \\
\hat{\gamma} & \pm t_{.99375}(n - p) \cdot \text{SE}(\hat{\gamma}) = 3.4542 \pm 2.8961(0.8093) = (1.11, 5.80).
\end{align*}
\]

• In MatLab, the Delta Method of computing standard errors for parameter estimates is the default technique and can be easily obtained after calling the `nlinfit` function, by typing:

\[
ci2 = \text{nlparci(betahat2, resid2, J2)} \quad \% \text{Computes NLS Delta Method CIs}
\]

This returns approximate 95% *individual* confidence intervals for the three model parameters based on the Delta Method standard errors.

• We will next explore the use of bootstrapping as an alternative to using Delta Method-based standard errors. Both techniques have their problems and consider parameter uncertainty for each parameter *separately* rather than jointly. Later in the course, we will briefly explore the use of Markov chain Monte Carlo (MCMC) methods which account for the dependence among model parameters in measuring uncertainty.