

Chapter 13

Introduction to Nonlinear Regression and Neural Networks

13.1 Linear and Nonlinear Regression Models

In previous chapters we discussed multiple regression models such as

$$Y_i = \beta_o + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_{p-1} X_{i(p-1)} + \epsilon_i$$

Or

$$Y_i = \beta_o + \beta_1 X_{i,1} + \beta_{11} X_{i,1}^2 + \beta_3 X_{i2} + \beta_4 X_{i2}^2 + \beta_5 X_{i1} X_{i2} + \epsilon_i$$

Or

$$\log(Y_i) = \beta_o + \beta_1 \sqrt{X_{i1}} + \beta_2 X_{i2} + \epsilon_i$$

In general

$$Y_i = f(X_i, \beta) + \epsilon_i$$

Nonlinear regression

Consider this model.

$$Y_i = f(X_i, \gamma) + \epsilon_i$$

where $f(X_i, \gamma)$ is nonlinear in γ . For example

$$Y_i = \gamma_o e^{\gamma_1 X_i} + \epsilon_i$$

or

$$Y_i = \log\left(\frac{\gamma_o + \gamma_1 X_i}{\gamma_o \gamma_1}\right) + \epsilon_i$$

Exponential Regression Model

$$Y_i = \gamma_o e^{\gamma_1 X_i} + \epsilon_i$$

where:

γ_0 and γ_1 are known parameters

X_i are known constants

ϵ_i are independent $N(0, \sigma^2)$

Logistic Regression Model

$$Y_i = \frac{\gamma_0}{1 + \gamma_1 \exp(\gamma_2 X_i)} + \epsilon_i$$

Estimation of Regression Parameters

In nonlinear regression, estimation of the parameters is still carried out using least squares or maximum likelihood. Unfortunately, it is usually impossible to find an equation for the form of the parameter estimate. Iterative methods must be used to find these parameter estimates.

13.2 Least Squares Estimation in Nonlinear Regression

We wish to minimize

$$Q = \sum_{i=1}^n (Y_i - f(X_i, \gamma))^2$$

The partial derivative of Q with respect to γ_k is

$$\frac{\partial Q}{\partial \gamma_k} = \sum_{i=1}^n -2[Y_i - f(X_i, \gamma)] \left[\frac{\partial f(X_i, \gamma)}{\partial \gamma_k} \right]$$

We now set all of these partial derivatives = 0 and obtain the normal equations

$$\sum_{i=1}^n Y_i \left[\frac{\partial f(X_i, \gamma)}{\partial \gamma_k} \right] - \sum_{i=1}^n f(X_i, \gamma) \left[\frac{\partial f(X_i, \gamma)}{\partial \gamma_k} \right] = 0$$

The problem with solving these equations is near impossible because these equations are nonlinear in γ . Hence, we will have to use numerical methods to solve the normal equations.

Direct Numerical Search - Gauss-Newton Method

The Gauss-Newton Method uses the Taylor series expansion to approximate the nonlinear regression model. This approximation simply uses the first linear term in the expansion to approximate the nonlinear function. We start with initial estimates of $\gamma_0, \gamma_1, \dots, \gamma_{p-1}$ with $g_0^{(0)}, g_1^{(0)}, \dots, g_{p-1}^{(0)}$.

$$f(X_i, \gamma) \approx f(X_i, g^{(0)}) + \sum_{k=0}^{p-1} \left[\frac{\partial f(X_i, \gamma)}{\partial \gamma_k} \right]_{\gamma=g^{(0)}} (\gamma_k - g_k^{(0)})$$

Call $g_k^{(0)}$ the initial estimate for the Taylor expansion of the k^{th} normal equation. Next,

$$f_i^{(0)} = f(X_i, \gamma_k^{(0)})$$

$$\beta_k^{(0)} = \gamma_k - g_k^{(0)}$$

$$D_{ik}^{(0)} = \left[\frac{\partial f(X_i, \gamma)}{\partial \gamma_k} \right]_{\gamma=g^{(0)}}$$

The Taylor approximation for the i^{th} case is then

$$f(X_i, \gamma) \approx f_i^{(0)} + \sum_{k=0}^{p-1} D_{ik}^{(0)} \beta_k^{(0)}$$

Note that we can then rewrite the regression equation as

$$Y_i \approx f_i^{(0)} + \sum_{k=0}^{p-1} D_{ik}^{(0)} \beta_k^{(0)} + \epsilon_i$$

Then, by moving $f_i^{(0)}$ to the other side of the equation, we get.

$$Y_i - f_i^{(0)} \approx \sum_{k=0}^{p-1} D_{ik}^{(0)} \beta_k^{(0)} + \epsilon_i$$

Next, call $Y_i^{(0)} = Y_i - f_i^{(0)}$. Then, we get

$$Y_i^{(0)} \approx \sum_{k=0}^{p-1} D_{ik}^{(0)} \beta_k^{(0)} + \epsilon_i$$

Note that we now have linear regression again. We estimate the $\beta_k^{(0)}$ s with $b_k^{(0)}$ s. Then,

$$g_k^{(1)} = g_k^{(0)} + b_k^{(0)}$$

The method starts all over until the minimum SSE is found.

Example

$$Y_i = \gamma_o e^{\gamma_1 X_i} + \epsilon_i$$

We wish to minimize

$$Q = \sum_{i=1}^n (Y_i - f(X_i, \gamma))^2 = \sum_{i=1}^n (Y_i - \gamma_o e^{\gamma_1 X_i})^2$$

Using the Gauss-Newton method we first need the partial derivative of $f(X, \gamma)$ with respect to γ_o

$$D_{io} = \frac{\partial \gamma_o e^{\gamma_1 X_i}}{\partial \gamma_o} = e^{\gamma_1 X_i}$$

Likewise, the partial derivative of $f(X, \gamma)$ with respect to γ_1 is

$$D_{i1} = \frac{\partial \gamma_o e^{\gamma_1 X_i}}{\partial \gamma_1} = \gamma_o X_i e^{\gamma_1 X_i}$$

The partial derivative of $f(X, \gamma)$ with respect to γ_0 is

$$D_{io} = \frac{\partial \gamma_o e^{\gamma_1 X_i}}{\partial \gamma_o} = e^{\gamma_1 X_i}$$

Next, we find $Y_i^{(0)} = Y_i - f_i^{(0)}$ using initial estimates $g_o^{(0)}$ and $g_1^{(0)}$. Thus,

$$Y_i^{(0)} = Y_i - g_o^{(0)} e^{g_1^{(0)} X_i}$$

And finally, we find the linear regression model which estimates

$$Y_i^{(0)} \approx \sum_{k=0}^{p-1} D_{ik}^{(0)} \beta_k^{(0)} + \epsilon_i$$

$Y^{(1)}, Y^{(2)}, \dots$ are found until the minimum SSE is obtained. What makes this simple is the fact that

$$SSE^{(0)} = \sum_{i=1}^n (Y_i^{(0)})^2$$

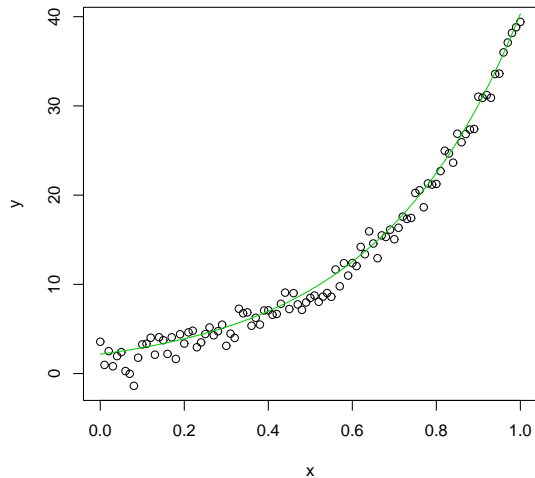
```

D_0 <- function(x,y,gamma_0,gamma_1){
  exp(gamma_1*x)
}
D_1 <- function(x,y,gamma_0,gamma_1){
  gamma_0*x*exp(gamma_1*x)
}
nonlinear_exp <- function(x,y,gamma_0,gamma_1,tol) {
  y_0 <- y-gamma_0*exp(gamma_1*x)
  SSE2 <- sum((y_0)^2)
  SSE1 <- SSE2+2*tol
  while (SSE2 < SSE1) {
    d_0 <- D_0(x,y,gamma_0,gamma_1)
    d_1 <- D_1(x,y,gamma_0,gamma_1)
    b_0 <- coef(lm(y_0~d_0+d_1))[2]
    b_1 <- coef(lm(y_0~d_0+d_1))[3]
    gamma_0 <- gamma_0+b_0
    gamma_1 <- gamma_1+b_1
    y_0 <- y-gamma_0*exp(gamma_1*x)
    SSE1 <- SSE2
    SSE2 <- sum((y_0)^2)
  }
  cat("The final SSE is",SSE2,"with parameter estimates of",gamma_0,"\n")
  cat("for gamma_0 and",gamma_1,"for gamma_1.\n")
  plot(x,y)
  lines(x,gamma_0*exp(gamma_1*x),col=3)
}

> nonlinear_exp(x,y,2,3,.01)
The final SSE is 134.1013 with parameter estimates of 2.174863
for gamma_0 and 2.919795 for gamma_1.
>
> nonlinear_exp(x,y,1,1,.01)
The final SSE is 53774946 with parameter estimates of 2.771117e-13
for gamma_0 and 37.50651 for gamma_1.
>

```

Notice the estimation procedure works quite well when the initial estimates are close to the true values but is terrible when the values are off somewhat. This is a problem with this method.



Here is an example of the method of steepest descent.

```

dgamma0 <- function(x,y,gamma0,gamma1){
  e <- exp(gamma1*x)
  -2*sum((y-gamma0*e)*e)
}

dgamma1 <- function(x,y,gamma0,gamma1){
  e <- gamma0*x*exp(gamma1*x)
  -2*sum((y-gamma0*exp(gamma1*x))*e)
}

unitgrad <- function(x,y,gamma0,gamma1){
  vec1 <- dgamma0(x,y,gamma0,gamma1)
  vec2 <- dgamma1(x,y,gamma0,gamma1)
  grad <- c(vec1,vec2)
  grad/sqrt(vec1^2+vec2^2)
}

thebigprogram <- function(x,y,gamma0,gamma1,alpha,tol){
  sse <- sum((y-gamma0*exp(x*gamma1))^2)
  c <- unitgrad(x,y,gamma0,gamma1)
  new_gamma0 <- gamma0-c[1]*alpha
  new_gamma1 <- gamma1-c[2]*alpha
  sse2 <- sum((y-new_gamma0*exp(x*new_gamma1))^2)
  while (sse-sse2 >tol) {
    c <- unitgrad(x,y,new_gamma0,new_gamma1)
  }
}

```

```

new_gamma0 <- new_gamma0-c[1]*alpha
new_gamma1 <- new_gamma1-c[2]*alpha
sse <- sse2
sse2 <- sum((y-new_gamma0*exp(x*new_gamma1))^2)
}
c(new_gamma0,new_gamma1,sse2)
}

x <- runif(25,5,10)
y <- 5*exp(2*x)+rnorm(25)*15

```

13.4 Inferences about Nonlinear Regression Parameters

Exact inference procedure are known for linear regression models with normal error terms. Unfortunately, this is not so for nonlinear models, regardless of the error terms. But, it is known that as the sample size goes up, the parameter estimates in most models start to have a “normal” looking distribution. We call this “Large-Sample Theory” or “Asymptotic Theory” Hence, we now approximate the sampling distribution of a parameter estimate with the t-distribution, just as in ordinary linear regression.

Estimate of Error Term Variance

$$MSE = \frac{\sum [Y_i - \hat{Y}_i]^2}{n - p}$$

Interval Estimation of a Single γ_k

When the error terms are $N(0, \sigma^2)$, then the expected value of $\hat{\gamma}$ is

$$E(\hat{\gamma}) \approx \gamma$$

where $\hat{\gamma}$ and γ are both vectors. Also, the standard deviation of $\hat{\gamma}$ is given by

$$s^2 \{\hat{\gamma}\} = MSE (D'D)^{-1}$$

where D is the matrix of partial derivatives evaluated at the final least squares estimate of γ

$$\frac{\hat{\gamma}_k - \gamma_k}{s\{\hat{\gamma}_k\}} \sim t(n - p)$$

Thus, a $(1 - \alpha) \times 100\%$ confidence interval for γ_k is given by

$$(\hat{\gamma}_k - t_{(\alpha/2, n-p)} s\{\hat{\gamma}_k\}, \hat{\gamma}_k + t_{(\alpha/2, n-p)} s\{\hat{\gamma}_k\})$$

Estimation of $s\{\hat{\gamma}_k\}$

When the error terms are not $N(0, \sigma^2)$ or if a more precise estimate of $s\{\hat{\gamma}_k\}$ is desired, then the bootstrap is one of the best means of estimating the standard deviation of the estimate $\hat{\gamma}_k$. Here is a sample program using the previous method of steepest descent program to find a bootstrap estimate for the standard deviation of γ .

```
boot_nonlinear <- function(x,y,gamma0,gamma1,alpha,tol,b){
  output <- thebigprogram(x,y,gamma0,gamma1,alpha,tol)
  gamma_0 <- output[1]
  gamma_1 <- output[2]
  residuals <- y-gamma_0*exp(gamma_1*x)
  boot_gamma_0 <- NULL
  boot_gamma_1 <- NULL
  for (i in 1:b) {
    newy <- gamma_0*exp(gamma_1*x)+sample(residuals,replace=T)
    boot_output <- thebigprogram(x,newy,gamma_0,gamma_1,alpha,tol)
    boot_gamma_0 <- c(boot_gamma_0,boot_output[1])
    boot_gamma_1 <- c(boot_gamma_1,boot_output[2])
  }
  cat("The bootstrap estimate of the standard deviation\n")
  cat("of the gamma_0 estimate is",sqrt(var(boot_gamma_0)),"\n")
  cat("The bootstrap estimate of the standard deviation\n")
  cat("of the gamma_1 estimate is",sqrt(var(boot_gamma_1)),"\n")
}
```

When is Large-Sample Theory Applicable

1. If you have a quick convergence in your iterative Gauss-Newton procedure, then this implies that the linearity approximation is a good fit.
2. If little skewness is found in your estimators, then a large-sample theory is appropriate.
3. If the parameter estimates appear “close-to-linear” or more accurately, “close-to-normal”, then large-sample theory is appropriate.