Lecture 10: Nonlinear Least Squares II

ResEcon 703: Topics in Advanced Econometrics

Matt Woerman
University of Massachusetts Amherst

Agenda

Last time

- Nonlinear Least Squares

Today

- Nonlinear Least Squares Example in R

Upcoming

- Reading for next time
  - Greene textbook, Chapters 13.1–13.5
- Problem sets
  - Problem Set 2 is posted, due October 17

Nonlinear Least Squares

$$y_i = h(x_i, \beta) + \varepsilon_i$$

The nonlinear least squares estimator minimizes the sum of squared errors for this model

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} [y_i - h(x_i, \beta)]^2$$

This estimator does not require a distributional assumption about our data (unlike MLE), but it has fewer nice properties

- Consistent
- Asymptotically normal
- Does not achieve the Cramer-Rao lower bound
- Does not have the “invariance” property

Nonlinear Least Squares Example in R

Multinomial Logit Estimation Example

We are again studying how consumers make choices about expensive and highly energy-consuming systems in their homes. We have data on 900 households in California and the type of heating system in their home. Each household has the following choice set, and we observe the following data

Choice set

- GC: gas central
- GR: gas room
- EC: electric central
- ER: electric room
- HP: heat pump

Alternative-specific data

- IC: installation cost
- OC: annual operating cost

Household demographic data

- income: annual income
- agehed: age of household head
- rooms: number of rooms
- region: home location

Load Dataset

```
### Load and look at dataset
## Load tidyverse and mlogit
library(tidyverse)
library(mlogit)
## Load dataset from mlogit package
data('Heating', package = 'mlogit')
```

Dataset

```
## Look at dataset
as_tibble(Heating)
## # A tibble: 900 x 16
##    idcase depvar ic.gc ic.gr ic.ec ic.er ic.hp oc.gc oc.gr oc.ec oc.er
##     <dbl> <fct>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1      1 gc       866  963.  860.  996. 1136.  200.  152.  553.  506.
##  2      2 gc       728.  759.  797.  895.  969.  169.  169.  520.  486.
##  3      3 gc       599.  783.  720.  900. 1048.  166.  138.  439.  405.
##  4      4 er       835.  793.  761.  831. 1049.  181.  147.  483   425.
##  5      5 er       756.  846.  859.  986.  883.  175.  139.  404.  390.
##  6      6 gc       666.  842.  694.  863.  859.  136.  141.  398.  371.
##  7      7 gc       670.  941.  634.  952. 1087.  192.  148.  478.  446.
##  8      8 gc       778. 1022.  813. 1012.  990.  188.  159.  502.  465.
##  9      9 gc       928. 1212.  876. 1025. 1232.  169.  190.  553.  452.
## 10     10 gc       683. 1045.  776.  874.  878.  176.  136.  532.  472.
## # ... with 890 more rows, and 5 more variables: oc.hp <dbl>,
## #   income <dbl>, agehed <dbl>, rooms <dbl>, region <fct>
```

Two Models to Estimate

Base model

$$U_{nj} = \alpha_j + \beta_1 IC_{nj} + \beta_2 OC_{nj} + \varepsilon_{nj}$$

Alternative-specific coefficients model

$$U_{nj} = \alpha_j + \beta_1 IC_{nj} + \beta_2 OC_{nj} + \beta_{3j} R_n + \varepsilon_{nj}$$

Optimization in R

```
### Optimization in R
## Help file for the optimization function, optim
?optim
## Arguments for optim function
optim(par, fn, gr, ..., method, lower, upper, control, hessian)
```

`optim()` requires that you create a function, `fn`, that

1. Takes a set of parameters and data as inputs
2. Calculates your objective function given those parameters
3. Returns this value of the objective function

You also have to give `optim()` arguments for

- `par`: starting parameter values
- `...`: dataset and other things needed by your function
- `method`: optimization algorithm
    - I recommend `method = 'BFGS'` for our estimation

Calculating the Sum of Squared Error in a Logit Model

Steps to calculate the sum of squared errors for a given set of parameters in a logit model

1. Calculate the representative utility for each alternative and for each decision maker.
2. Calculate the choice probability for each alternative and for each decision maker.
3. Calculate the econometric residual, or the difference between the outcome and the probability, for each alternative and for each decision maker.
4. Sum the square of these residuals.
5. Return the sum of squares.

Base Model

Random utility model: $U_{nj} = \alpha_j + \beta_1 IC_{nj} + \beta_2 OC_{nj} + \varepsilon_{nj}$

NLS regression model: $y_{ni} = \dfrac{e^{\alpha_i + \beta_1 IC_{ni} + \beta_2 OC_{ni}}}{\sum_{j=1}^{J} e^{\alpha_j + \beta_1 IC_{nj} + \beta_2 OC_{nj}}} + \omega_{ni}$

Function to Calculate Sum of Squares

```
### Calculate NLS estimator for multinomial logit heating choice using cost data
### and alternative effects
## Function to calculate sum of squares using two variables and four
## alternative effects
least_squares_long <- function(parameters, data){
  ## Extract individual parameters
  alphas <- parameters[1:4]
  beta_1 <- parameters[5]
  beta_2 <- parameters[6]
  ## Assign constant parameters to alternatives
  data <- data %>%
    group_by(idcase) %>%
    arrange(idcase, alt) %>%
    mutate(constant = c(alphas, 0)) %>%
    ungroup()
  ## Calculate utility for each alternative given the parameters
  data <- data %>%
    mutate(utility = constant + beta_1 * var_1 + beta_2 * var_2)
  ## Calculate logit probability denominator given the parameters
  data <- data %>%
    group_by(idcase) %>%
    mutate(probability_denominator = sum(exp(utility))) %>%
    ungroup()
  ## Calculate logit choice probability given the parameters
  data <- data %>%
    mutate(probability = exp(utility) / probability_denominator)
  ## Calculate regression residual given the parameters
  data <- data %>%
    mutate(residual = choice - probability)
  ## Calculate the sum of squares given the parameters
  sum_squares <- sum(data$residual^2)
  return(sum_squares)
}
```

Estimate NLS Parameters

```
## Gather heating dataset into a long dataset
heating_long <- Heating %>%
  gather(key, value, starts_with('ic.'), starts_with('oc.')) %>%
  separate(key, c('cost', 'alt')) %>%
  spread(cost, value) %>%
  mutate(choice = (depvar == alt)) %>%
  select(-depvar) %>%
  mutate(var_1 = ic, var_2 = oc)
## Minimize the sum of squared errors
model_nls_long <- optim(rep(0, 6), least_squares_long,
                        data = heating_long, method = 'BFGS')
```

Estimation Results

```
## Show NLS estimation results
model_nls_long
## $par
## [1]  1.862237887  2.048949391  1.561910402  0.127548314 -0.001782635
## [6] -0.008059906
##
## $value
## [1] 496.7162
##
## $counts
## function gradient
##      268       36
##
## $convergence
## [1] 0
##
## $message
## NULL
```

NLS Variance-Covariance Matrix Estimator

$$\widehat{Var}(\hat{\beta}) = \hat{\sigma}^2 \left[\sum_{i=1}^{n} \left(\frac{\partial h(x_i, \beta)}{\partial \beta} \right)_{\hat{\beta}} \left(\frac{\partial h(x_i, \beta)}{\partial \beta'} \right)_{\hat{\beta}} \right]^{-1}$$

Steps to estimate this variance-covariance matrix

1. Write down the derivative of the nonlinear regression model with respect to each of the K parameters.
2. Calculate this $K \times 1$ vector of derivatives, at the estimated parameters, for each decision maker.
3. Calculate the $K \times K$ matrix that is the product of the above vector and its transpose for each decision maker.
4. Sum these matrices for all decision makers.
5. Estimate the variance of the econometric error as the mean sum of squares at the estimated parameters.
6. Calculate the variance-covariance matrix, which is a function of the above $K \times K$ matrix and the estimated error variance.

Gradient of Logit Choice Probabilities

$$\frac{\partial P_{ni}}{\partial \alpha_i} = P_{ni}(1 - P_{ni})$$

$$\frac{\partial P_{ni}}{\partial \alpha_{j|j \neq i}} = -P_{ni}P_{nj}$$

$$\frac{\partial P_{ni}}{\partial \beta_1} = P_{ni}(IC_{ni} - \sum_{j=1}^{J} P_{nj} IC_{nj})$$

$$\frac{\partial P_{ni}}{\partial \beta_2} = P_{ni}(OC_{ni} - \sum_{j=1}^{J} P_{nj} OC_{nj})$$

Estimating the NLS Variance-Covariance Matrix

```
### Estimate the variance of the previous NLS model
## Assign constant parameters to alternatives
variance_data <- heating_long %>%
  group_by(idcase) %>%
  arrange(idcase, alt) %>%
  mutate(constant = c(model_nls_long$par[1:4], 0)) %>%
  ungroup()
## Calculate utility for each alternative at the NLS parameters
variance_data <- variance_data %>%
  mutate(utility = constant +
           model_nls_long$par[5] * var_1 + model_nls_long$par[6] * var_2)
## Calculate logit probability denominator at the NLS parameters
variance_data <- variance_data %>%
  group_by(idcase) %>%
  mutate(probability_denominator = sum(exp(utility))) %>%
  ungroup()
## Calculate logit choice probability at the NLS parameters
variance_data <- variance_data %>%
  mutate(probability = exp(utility) / probability_denominator)
## Create vectors of individual probabilities
variance_data <- variance_data %>%
  select(idcase, alt, probability) %>%
  spread(alt, probability) %>%
  mutate(probability_all = pmap(list(.$ec, .$er, .$gc, .$er),
                              ~ c(..1, ..2, ..3, ..4))) %>%
  select(idcase, probability_all) %>%
  full_join(variance_data, by = 'idcase')
```

Estimating the NLS Variance-Covariance Matrix

```
## Calculate the probability-weighted average of each cost for each individual
variance_data <- variance_data %>%
  group_by(idcase) %>%
  mutate(ic_weighted = sum(probability * ic),
         oc_weighted = sum(probability * oc)) %>%
  ungroup()
## Calculate the gradient for each alternative and individual
variance_data <- variance_data %>%
  group_by(idcase) %>%
  arrange(idcase, alt) %>%
  mutate(alt_order = 1:n()) %>%
  ungroup() %>%
  mutate(constant_vector = map(.$alt_order, ~ c(rep(0, .x - 1),
                                                1,
                                                rep(0, 5 - .x))[1:4])) %>%
  mutate(gradient_alpha = pmap(list(.$probability,
                                    .$probability_all,
                                    .$constant_vector),
                               ~ ..1 * (..3 - ..2))) %>%
  mutate(gradient = pmap(list(.$gradient_alpha, .$probability,
                              .$ic, .$ic_weighted, .$oc, .$oc_weighted),
                         ~ c(..1,
                             ..2 * (..3 - ..4),
                             ..2 * (..5 - ..6))))
## Calculate the gradient product matrix for each alternative and individual
variance_data <- variance_data %>%
  mutate(gradient_matrix = map(.$gradient, ~ .x %*% t(.x)))
## Sum the gradient product matrices over all alternatives and individuals
gradient_matrix_sum <- variance_data$gradient_matrix %>% reduce(`+`)
## Estimate the variance of the econometric errors
error_variance <- least_squares_long(model_nls_long$par, heating_long) /
  nrow(heating_long)
```

NLS Variance-Covariance Matrix

```
## Calculate the variance-covariance matrix
variance_covariance <- error_variance * solve(gradient_matrix_sum)
variance_covariance
##               [,1]          [,2]         [,3]          [,4]
## [1,]  1.415897e-01  9.891294e-02 1.383776e-02 -1.377707e-03
## [2,]  9.891294e-02  9.403330e-02 8.352834e-03 -9.076836e-04
## [3,]  1.383776e-02  8.352834e-03 3.131992e-02  2.679804e-02
## [4,] -1.377707e-03 -9.076836e-04 2.679804e-02  3.308934e-02
## [5,]  5.294321e-05  1.959156e-05 4.990586e-05  2.008979e-05
## [6,] -4.093777e-04 -3.346047e-04 7.533308e-05  1.170280e-04
##               [,5]          [,6]
## [1,]  5.294321e-05 -4.093777e-04
## [2,]  1.959156e-05 -3.346047e-04
## [3,]  4.990586e-05  7.533308e-05
## [4,]  2.008979e-05  1.170280e-04
## [5,]  2.252765e-07 -3.809546e-08
## [6,] -3.809546e-08  1.838542e-06
```

NLS Parameters and Standard Errors

```
## Report estimated parameters and standard errors
model_nls_long$par
## [1]  1.862237887  2.048949391  1.561910402  0.127548314 -0.001782635
## [6] -0.008059906

variance_covariance %>%
  diag() %>%
  sqrt()
## [1] 0.376284032 0.306648496 0.176974339 0.181904744 0.000474633
## [6] 0.001355928
```

NLS Hypothesis Test

Test that the alternative-specific intercepts are jointly significant

$$H_0 : \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

NLS Wald Test

Hypothesis: $H_0\colon\ r(\beta_0) = q$

Test statistic: $W = [r(\hat{\beta}) - q]'[R(\hat{\beta})\widehat{Var}(\hat{\beta})R'(\hat{\beta})]^{-1}[r(\hat{\beta}) - q]$

Jacobian matrix: $R(\hat{\beta}) = \left(\dfrac{\partial r(\beta)}{\partial \beta'}\right)_{\hat{\beta}}$

Steps to conduct a Wald Test using NLS estimators

1. Create a vector of J parameter restrictions, $r(\beta_0) = q$.
2. Calculate the $J \times K$ Jacobian matrix by differentiating each restriction with respect to each of the K parameters.
3. Calculate the Wald test statistic, which is a function of the vector of restrictions, the Jacobian matrix, and the variance-covariance matrix.
4. Conduct the Wald test using this test statistic, which is distributed χ^2.

Calculating the NLS Wald Test Statistic

```r
### Conduct a Wald test that alternative-specific intercepts equal zero
## Calculate the restriction vector
restriction_vector <- model_nls_long$par[1:4]
## Construct the restriction Jacobian
restriction_jacobian <- diag(4) %>%
  cbind(rep(0, 4)) %>%
  cbind(rep(0, 4))
## Calculate the Wald test statistic
wald_test_stat <- t(restriction_vector) %*%
  solve(restriction_jacobian %*%
          variance_covariance %*%
          t(restriction_jacobian)) %*%
  restriction_vector %>%
  c()
```

NLS Wald Test Results

```
## Test if Wald test statitsic is greater than critical value
wald_test_stat > qchisq(0.95, 4)
## [1] TRUE

## Calculate p-value of Wald test
1 - pchisq(wald_test_stat, 4)
## [1] 0
```

Alternative-Specific Coefficients Model

Random utility model: $U_{nj} = \alpha_j + \beta_1 IC_{nj} + \beta_2 OC_{nj} + \beta_{3j} R_n + \varepsilon_{nj}$

NLS regression model: $y_{ni} = \dfrac{e^{\alpha_i + \beta_1 IC_{ni} + \beta_2 OC_{ni} + \beta_{3j} R_n}}{\sum_{j=1}^{J} e^{\alpha_j + \beta_1 IC_{nj} + \beta_2 OC_{nj} + \beta_{3j} R_n}} + \omega_{ni}$

Alternative-Specific Coefficients Estimation

```
### Calculate NLS estimator for multinomial logit heating choice using cost
### data, alternative effects, and alternative-specific coefficients on rooms
## Function to calculate sum of squares using flexible matrices
least_squares_matrix <- function(parameters, data){
  ## Extract explanatory variables
  data_x <- data %>%
    map(1)
  ## Extract choice data
  data_y <- data %>%
    map(2)
  ## Calculate utility for each alternative given the parameters
  utility <- data_x %>%
    map(~ .x %*% parameters)
  ## Calculate logit probability denominator given the parameters
  probability_denominator <- utility %>%
    map(~ sum(exp(.x)))
  ## Calculate logit probability numerator given the parameters
  probability_numerator <- utility %>%
    map(~ exp(.x))
  ## Calculate logit choice probability given the parameters
  probability_choice <- probability_numerator %>%
    map2(probability_denominator, ~ .x / .y)
  ## Calculate square residual given the parameters
  residuals <- data_y %>%
    map2(probability_choice, ~ .x - .y)
  ## Calculate the sum of squares given the parameters
  sum_squares <- residuals %>%
    map(~ sum(.x^2)) %>%
    unlist() %>%
    sum()
  return(sum_squares)
}
```

Alternative-Specific Coefficients Estimation Data

```
## Split heating dataset into list of household data frames
heating_split <- heating_long %>%
  group_by(idcase) %>%
  arrange(idcase, alt) %>%
  group_split()
## Crate matrix of dummy variables (intercepts) to bind to data
constant_matrix <- diag(4) %>%
  rbind(c(0, 0, 0, 0))
## Function to create list of datasets for estimation
create_data_matrix <- function(data){
  data_x <- data %>%
    select(ic, oc) %>%
    as.matrix()
  data_x <- constant_matrix %>%
    cbind(data_x)
  rooms <- data$rooms[1]
  data_x <-
    data_x %>%
    cbind(rooms * constant_matrix)
  data_y <- data %>%
    select(choice) %>%
    mutate(choice = 1 * choice) %>%
    pull(choice)
  return(list(x = data_x, y = data_y))
}
## Create list of datasets for estimation
heating_matrix <- heating_split %>%
  map(.x = ., .f = ~ create_data_matrix(.x))
```

Alternative-Specific Coefficients Optimization

```r
## Minimize the sum of square errors
model_matrix <- optim(rep(0, 10), least_squares_matrix,
                      data = heating_matrix, method = 'BFGS')
```

Alternative-Specific Coefficients NLS Results

```
## Show NLS parameters
model_matrix$par
##  [1]  1.738552715  2.198846013  1.666259660  0.334077782 -0.001848879
##  [6] -0.008319251  0.030034988 -0.030596610 -0.031776661 -0.054900639
```

Announcements

Reading for next time

- Greene textbook, Chapters 13.1–13.5

Office hours

- Reminder: Tuesdays at 2:00–3:00 in 218 Stockbridge

Upcoming

- Problem Set 2 is posted, due October 17