w271: Homework 4 (Due: Week 5)

Professor Jeffrey Yau

Due: 4pm Pacific Time on the Day of the Live Session of Week 5

Instructions (Please Read it Carefully!):

- **Page limit of the pdf report: None, but please be reasonable**
- Page setup:
- Use the following font size, margin, and linespace:
  - fontsize=11pt
  - margin=1in
  - line_spacing=single
- Submission:
  - Each student submits his/her homework to the course github repo by the deadline; submission and revision made after the deadline will not be graded
  - Submit 2 files:
    1. A pdf file that details your answers. Include all the R codes used to produce the answers. *Please do not suppress the codes in your pdf file.*
    2. R markdown file used to produce the pdf file
  - Use the following file-naming convensation; fail to do so will receive 10% reduction in the grade:
    - StudentFirstNameLastName_HWNumber.fileExtension
    - For example, if the student's name is Kyle Cartman for homework 1, name your files as
      - KyleCartman_HW1.Rmd
      - KyleCartman_HW1.pdf
  - Although it sounds obvious, please write your name on page 1 of your pdf and Rmd files.
  - For statistical methods that we cover in this course, use only the R libraries and functions that are covered in this course. If you use libraries and functions for statistical modeling that we have not covered, you have to (1) provide an explanation of why such libraries and functions are used instead and (2) reference to the library documentation. **Lacking the explanation and reference to the documentation will result in a score of zero for the corresponding question.** For data wrangling and data visualization, you are free to use other libraries, such as dplyr, ggplot2, etc.
- For mathematical formulae, type them in your R markdown file. **Do not write them on a piece of paper, snap a photo, and either insert the image file or sumbit the image file separately. Doing so will receive a** 0 **for that whole question.**

- Students are expected to act with regards to UC Berkeley Academic Integrity.

*

*Question 18 a and b of Chapter 3 (page 192,193)**

For the wheat kernel data (*wheat.csv*), consider a model to estimate the `kernel condition` using the `density` explanatory variable as a linear term.

Part A

a. Write an R function that computes the log-likelihood function for the multinomial regression model. Evaluate the function at the parameter estimates produced by `multinom()`, and verify that your computed value is the same as that produced by `logLik()` (use the object saved from `multinom()` within this function).

Answer:

We are going to answer this question in a few steps.

1. Set up the some initial codes and loading the data.
2. Estimate the model whose specification is given by the question using the `multinom()` function and obtain the model object's components.
3. Write a custom R function that computes the log-likelihood function for the multinomial regression model.
4. Evaluate the function at the parameter estimates produced by `multinom()`, and verify that the computed value is the same as that produced by `logLik()` (use the object saved from `multinom()` within this function).

1. Set up the some initial codes and loading the data.

```
rm(list = ls())

knitr::opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)

# Load Libraries
library(car)
library(Hmisc)
# library(skimr)
library(ggplot2)
library(stargazer)
# library(gmodels) # For cross tabulation (SAS and SPSS style)

library(MASS)
library(mcprofile)
# library(vcd)
library(nnet)

wheat <- read.csv("../wheat.csv", stringsAsFactors = FALSE, header = TRUE, sep = ",")

str(wheat)
```

```
## 'data.frame':    275 obs. of  7 variables:
##  $ class   : chr  "hrw" "hrw" "hrw" "hrw" ...
##  $ density : num  1.35 1.29 1.23 1.34 1.26 ...
##  $ hardness: num  60.3 56.1 44 53.8 44.4 ...
##  $ size    : num  2.3 2.73 2.51 2.27 2.35 ...
##  $ weight  : num  24.6 33.3 31.8 32.7 26.1 ...
##  $ moisture: num  12 12.2 11.9 12.1 12.1 ...
##  $ type    : chr  "Healthy" "Healthy" "Healthy" "Healthy" ...
```

2. Estimate the model whose specification is given by the question using thge `multinom()` function and obtain the model object's components.

```
# model
nominal.fit1 <- multinom(as.factor(type) ~ density, data = wheat)
```

```
## # weights:  9 (4 variable)
## initial  value 302.118379
## iter  10 value 229.769334
## iter  20 value 229.712304
## final  value 229.712290
## converged
```

```
summary(nominal.fit1)
```

```
## Call:
## multinom(formula = as.factor(type) ~ density, data = wheat)
##
## Coefficients:
##        (Intercept)   density
## Scab      29.37827 -24.56215
## Sprout    19.12165 -15.47633
##
## Std. Errors:
##        (Intercept)  density
## Scab      3.676892 3.017842
## Sprout    3.337092 2.691429
##
## Residual Deviance: 459.4246
## AIC: 467.4246
```

```
# log likelihood of fitted model
logLik(nominal.fit1)
```

```
## 'log Lik.' -229.7123 (df=4)
```

```
# variance / co-variance matrix
round(vcov(nominal.fit1), 2)
```

```
##                     Scab:(Intercept) Scab:density Sprout:(Intercept)
## Scab:(Intercept)               13.52       -11.08               10.33
## Scab:density                  -11.08         9.11               -8.33
```

```
## Sprout:(Intercept)            10.33        -8.33              11.14
## Sprout:density                -8.27         6.68              -8.97
##                     Sprout:density
## Scab:(Intercept)            -8.27
## Scab:density                 6.68
## Sprout:(Intercept)          -8.97
## Sprout:density               7.24
```

3. Write a custom R function that computes the log-likelihood function for the multinomial regression model. 4. Evaluate the function at the parameter estimates produced by `multinom()`, and verify that the computed value is the same as that produced by `logLik()` (use the object saved from `multinom()` within this function).

```
compute.logL <- function(beta, x, y) {
    # reference pg. 152 of textbook
    den <- 1 + exp(beta[1] + beta[3] * x) + exp(beta[2] + beta[4] *
        x)
    pi1 <- 1/den
    pi2 <- exp(beta[1] + beta[3] * x)/den
    pi3 <- exp(beta[2] + beta[4] * x)/den
    LogL <- sum(y[, 1] * log(pi1) + y[, 2] * log(pi2) + y[, 3] *
        log(pi3))
    return(LogL)
}

# Data
healthy <- ifelse(test = wheat$type == "Healthy", yes = 1, no = 0)
sprout <- ifelse(test = wheat$type == "Sprout", yes = 1, no = 0)
scab <- ifelse(test = wheat$type == "Scab", yes = 1, no = 0)
y <- cbind(healthy, scab, sprout)
x <- wheat$density
```

4. Evaluate the function at the parameter estimates produced by `multinom()`, and verify that the computed value is the same as that produced by `logLik()` (use the object saved from `multinom()` within this function).

```
# Parameters estimates from multinom() are in beta.hat
beta.hat <- coef(nominal.fit1)
logL <- compute.logL(beta = beta.hat, x = x, y = y)
logL
```

```
## [1] -229.7123
```

Part B

b. Maximize the log-likelihood function using `optim()` to obtain the MLEs and the estimated covariance matrix. Compare your answers to what is obtained by `multinom()`.

Note that to obtain starting values for `optim()`, one approach is to estimate separate logistic

regression models for $log\left(\frac{\pi_2}{\pi_1}\right)$ and $log\left(\frac{\pi_3}{\pi_1}\right)$. These models are estimated only for those observations that have the corresponding responses (e.g., a $Y = 1$ or $Y = 2$ for $log\left(\frac{\pi_2}{\pi_1}\right)$).

Answers:

We will just initialize paramters to random instead of computing logistic regression models as the question suggests.

```
# Data
healthy <- ifelse(test = wheat$type == "Healthy", yes = 1, no = 0)
sprout <- ifelse(test = wheat$type == "Sprout", yes = 1, no = 0)
scab <- ifelse(test = wheat$type == "Scab", yes = 1, no = 0)
y <- cbind(healthy, scab, sprout)
x <- as.matrix(wheat$density)

# Use the optimization routine to find the estimate
opt <- optim(rnorm(4), compute.logL, x = x, y = y, control = list(fnscale = -1),
    hessian = TRUE)

# hessian matrix
hessian <- opt$hessian

# paramter updates
pars <- data.frame(matrix(opt$par, ncol = 2), row.names = c("Scab",
    "Sprout"))
colnames(pars) <- c("Intercept", "density")

pars
```

```
##        Intercept   density
## Scab    29.38423 -24.56665
## Sprout  19.12584 -15.47948
```

```
hessian
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] -39.36995  27.48653 -45.59217  31.13374
## [2,]  27.48653 -57.76681  31.13374 -68.86732
## [3,] -45.59217  31.13374 -53.15003  35.51616
## [4,]  31.13374 -68.86732  35.51616 -82.62956
```

Compute the variance / co-variance matrix using Hessian matrix output...

```
vcov <- data.frame(-solve(hessian), row.names = c("Scab:Intercept",
    "Sprout:Intercept", "Scab:density", "Sprout:density"))

colnames(vcov) <- c("Scab:Intercept", "Sprout:Intercept", "Scab:density",
    "Sprout:density")
vcov
```

```
##                  Scab:Intercept Sprout:Intercept Scab:density
```

```
## Scab:Intercept        13.524253        10.329093   -11.080687
## Sprout:Intercept      10.329093        11.140259    -8.331264
## Scab:density         -11.080687        -8.331264     9.110344
## Sprout:density        -8.275728        -8.973919     6.684448
##                   Sprout:density
## Scab:Intercept        -8.275728
## Sprout:Intercept      -8.973919
## Scab:density           6.684448
## Sprout:density         7.246333
```

.. as compared to the vcov output from multinom()

```
round(vcov(nominal.fit1), 2)
```

```
##                     Scab:(Intercept) Scab:density Sprout:(Intercept)
## Scab:(Intercept)                13.52       -11.08                10.33
## Scab:density                   -11.08         9.11                -8.33
## Sprout:(Intercept)              10.33        -8.33                11.14
## Sprout:density                  -8.27         6.68                -8.97
##                     Sprout:density
## Scab:(Intercept)             -8.27
## Scab:density                  6.68
## Sprout:(Intercept)           -8.97
## Sprout:density                7.24
```