STA261 Week-12

Shahriar Shams

30/03/2020

Review of week-11

- Part-1 of lecture: correlation and Least square regression
  - Relationship among quantitative variables
  - Pearson correlation coefficient
  - Least square regression
- Part-2 of lecture: Regression under Normal distribution
  - Properties of estimators of regression parameters
  - Confidence interval/t-test for β_2
  - Sum of squares decomposition/ANOVA test

Learning goals

- Regression: Quantitative Y, Categorical X
- Computational aspects of likelihood estimation
- Newton-Raphson and Fisher's Scoring method (numerical methods)

Regression: Quantitative Y, Categorical X

- Assume we have response variable Y which is quantitative.
- And we have predictor X which is categorical
- We want to check whether X and Y are related or not.
- Let's assume a simple case where X only has two categories. (For example, male and female)
- We can create what's known as **dummy variables**.
- Let, $X_m = 1,$ if Male and $X_m = 0,$ if Female

- A hypothetical data will look like this:

Y	Sex (X)	X_m
10	Male	1
12	Male	1
8	Female	0
9	Female	0
...	...	...
...	...	...

- X_m is the numerical representation of the categorical variable Sex.
- Recall the Normality assumption from previous week.
  - we can write, $Y|X \sim N(\beta_1 + \beta_2 X_m,\ \sigma^2)$
  - Therefore, $E[Y|X] = \beta_1 + \beta_2 X_m$
  - $E[Y|X = Female] = \beta_1$
  - $E[Y|X = Male] = \beta_1 + \beta_2$

- Subtracting one from the other, we get

$$\beta_2 = E[Y|X = Male] - E[Y|X = Female]$$

- If we want to test whether the two group averages are equal or not that's same as testing $H_0 : \beta_2 = 0$
- Likelihood contribution of each of the y_i's with $X = 0$ will be

$$(2\pi\sigma^2)^{-1/2} exp[-\frac{1}{2\sigma^2}(y_i - \beta_1)^2]$$

- Likelihood contribution of each of the y_i's with $X = 1$ will be

$$(2\pi\sigma^2)^{-1/2} exp[-\frac{1}{2\sigma^2}(y_i - \beta_1 - \beta_2)^2]$$

- We can maximize the entire likelihood and calculate the estimates of β_1 and β_2
  - In this example(try it on your own),

$$\hat{\beta}_1 = \bar{y}_{\text{ for the female group}}$$

$$\hat{\beta}_2 = \bar{y}_{\text{ for the male group}} - \bar{y}_{\text{ for the female group}}$$

A numerical example

This example is taken from page 423 (example A) of the Rice text book.

```r
# Let's enter the data in R
G1=c(79.98,80.04,80.02,80.04,80.03,80.03,80.04,79.97,
     80.05,80.03,80.02,80.00,80.02)
G2=c(80.02,79.94,79.98,79.97,79.97,80.03,79.95,79.97)

mean(G1)
```

```
## [1] 80.02077
```

```r
mean(G2)
```

```
## [1] 79.97875
```

```r
# Converting into what is known as "long" format data
Y=c(G1,G2)
X_G1=c(rep(1,length(G1)),rep(0,length(G2)))

# Let's see how the data looks
cbind(Y,X_G1)
```

```
##            Y X_G1
##  [1,] 79.98    1
##  [2,] 80.04    1
##  [3,] 80.02    1
##  [4,] 80.04    1
##  [5,] 80.03    1
##  [6,] 80.03    1
##  [7,] 80.04    1
##  [8,] 79.97    1
##  [9,] 80.05    1
## [10,] 80.03    1
## [11,] 80.02    1
## [12,] 80.00    1
## [13,] 80.02    1
## [14,] 80.02    0
## [15,] 79.94    0
## [16,] 79.98    0
## [17,] 79.97    0
## [18,] 79.97    0
## [19,] 80.03    0
## [20,] 79.95    0
## [21,] 79.97    0
```

```r
# Fitting a linear model
m=lm(Y~X_G1)
summary(m)
```

```
##
## Call:
## lm(formula = Y ~ X_G1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.050769 -0.008750 -0.000769  0.019231  0.051250
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 79.978750   0.009521 8399.914  < 2e-16 ***
## X_G1         0.042019   0.012101    3.472  0.00255 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02693 on 19 degrees of freedom
## Multiple R-squared:  0.3882, Adjusted R-squared:  0.356
## F-statistic: 12.06 on 1 and 19 DF,  p-value: 0.002551
```

```r
# Confidence intervals of regression parameters
confint(m,level = 0.95)
```

```
##                   2.5 %      97.5 %
## (Intercept) 79.95882153 79.99867847
## X_G1         0.01669058  0.06734788
```

Let's do a t-test that we learned in Week-8 (slides 14 and 15)

```r
t.test(G1,G2, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  G1 and G2
## t = 3.4722, df = 19, p-value = 0.002551
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.01669058 0.06734788
## sample estimates:
## mean of x mean of y
##  80.02077  79.97875
```

- Will we be able to use this if our X variable has more than two categories?
  - Absolutely!
  - We will just need more dummy variables.
  - For example, if we have a variable called *program of study* which has three categories (Undergrad, Masters and PhD) we will need two dummy variables.

Program (X)	X_u	X_m
Undergrad	1	0
Masters	0	1
PhD	0	0

- But then we have multiple linear regression which you will learn in STA302, STA303 or similar courses.

Computational aspects of likelihood function

Likelihood maximization using R

- Let's use an old example (week-2, slides 14-16)
- Say we tossed a coin with $P[H] = \theta$ 5 times. And the outcomes are HTHHT. θ is the unknown parameter.
- We know the likelihood function is

$$L(\theta) = \theta * (1-\theta) * \theta * \theta * (1-\theta) = \theta^3(1-\theta)^2$$

- By plugging in different value of θ we can get different value of $L(\theta)$
  - For example, for $L(0.6) = 0.6 * 0.4 * 0.6 * 0.6 * 0.4$
- Let's use R to do this directly

```
x=c(1,0,1,1,0)

# individual probabilities (in continuous case these will be densities)
dbinom(x,size=1,prob=0.6)
```

```
## [1] 0.6 0.4 0.6 0.6 0.4
```

```
# Likelihood a product of these numbers
L_theta = prod( dbinom(x,size=1,prob=0.6) )
L_theta
```

```
## [1] 0.03456
```

- We can also calculate log-likelihoods directly

```
l_theta = sum( dbinom(x,size=1,prob=0.6,log=TRUE) )
l_theta
```

```
## [1] -3.365058
```

- We can use the *optim*() function in R to maximize the log-likelihood (or the likelihood) function.

```
# define l_theta as a function of the unknown parameter
l_theta = function(t){sum( dbinom(x,size=1,prob=t,log=TRUE) )}

# optim(initial guess, function to maximize, ... )
optim(0.001, l_theta, control=list(fnscale=-1) )
```

```
## $par
## [1] 0.5999
##
## $value
## [1] -3.365058
##
## $counts
## function gradient
##       46       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

- By default *optim*() minimizes a function.
  - $control = list(fnscale = -1)$ forces it to maximize the function
  - (I think) It still minimizes, but it minimizes $-1 * function$.

- We can get the second derivative(calculated at the MLE directly) from R as well

```
# hessian=TRUE option will calculate the second derivative
optim(0.001, l_theta, control=list(fnscale=-1) ,hessian=TRUE)
```

```
## $par
## [1] 0.5999
##
## $value
```

```
## [1] -3.365058
##
## $counts
## function gradient
##       46       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##           [,1]
## [1,] -20.83007
```

- Let's take a look at a more complex likelihood function. (week 9, slides 17-20)
- We looked at a LRT where we have data from two normal populations and we wanted to test $H_0 : \mu_x = \mu_y$ with the variances known.
- The numerical example that we did,

$(16.27, 11.66, 14.05, 15.43, 18.74, 13.42, 17.39, 18.71, 11.18, 13.52, 16.74, 5.43, 16.45, 10.75, 19.06) \sim N(\mu_x, \sigma_x = 3)$

$(10.89, 7.57, 15.39, 8.43, 12.33, 7.43, 5.56, 18.07, 0.35, 7.62) \sim N(\mu_y, \sigma_y = 4)$

- Test statistic,

$$-2ln\Lambda = -2ln\frac{L(\hat{\mu})}{L(\hat{\mu}_x, \hat{\mu}_y)}$$

```r
x=c(16.27, 11.66, 14.05, 15.43, 18.74, 13.42, 17.39, 18.71, 11.18,
    13.52, 16.74, 5.43,  16.45, 10.75, 19.06)
y=c(10.89, 7.57, 15.39, 8.43, 12.33, 7.43, 5.56, 18.07, 0.35, 7.62)

l_mu=function(mu){
  sum(dnorm(x,mean=mu,sd=3,log=TRUE))+sum(dnorm(y,mean=mu,sd=4,log=TRUE))
  }

optim(10, fn=l_mu, control=list(fnscale=-1))
```

```
## $par
## [1] 13.16211
```

```
##
## $value
## [1] -77.54672
##
## $counts
## function gradient
##       26       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```r
# This is the value of L(mu hat) that you saw on slide 20
exp(-77.54672)
```

```
## [1] 2.098396e-34
```

```r
l_2_mu=function(mu){
  sum(dnorm(x,mean=mu[1],sd=3,log=TRUE))+sum(dnorm(y,mean=mu[2],sd=4,log=TRUE))
  }

optim(c(10,10), fn=l_2_mu, control=list(fnscale=-1))
```

```
## $par
## [1] 14.587248  9.363925
##
## $value
## [1] -71.34758
##
## $counts
## function gradient
##       61       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```r
# This is the value of L(mu_x hat and mu_y hat) that you saw on slide 20
exp(-71.34758)
```

```
## [1] 1.033094e-31
```

- Maximizing this likelihood with considering the variances to be unknown as well

```
l_all_unknown=function(mu){
  sum(dnorm(x,mean=mu[1],sd=mu[3],log=TRUE))+
    sum(dnorm(y,mean=mu[2],sd=mu[4],log=TRUE))
  }

optim(c(10,10,2,2), fn=l_all_unknown, control=list(fnscale=-1))
```

```
## $par
## [1] 14.589693  9.360240  3.606365  4.791184
##
## $value
## [1] -70.39576
##
## $counts
## function gradient
##      139       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Newton-Raphson and Fisher's Scoring method (numerical methods)

Newton-Raphson (N-R)

- It's a very powerful way of finding solution of a equation numerically.
- Suppose we want to find the solution of $f(x) = 0$
- We can make an initial guess and say a is the solution of this equation.
- According to N-R, An updated solution (say b) is

$$b = a - \frac{f(a)}{f'(a)}$$

- We can use this to find the solution iteratively

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

- We can set up a stopping rule. Say if $|x_{n+1} - x_n| < 0.00001$ we will stop updating and take x_{n+1} as the solution of $f(x) = 0$
- A nice detailed reading on N-R can be found here https://www.math.ubc.ca/~anstee/math104/newtonmethod.pdf

- Let's look at an example and see how we can implement this in the likelihood maximization problem.
- Suppose, $X_1, X_2, ..., X_m \sim Exp(\theta)$
- Therefore,
  - $L(\theta) = (\frac{1}{\theta})^m exp[-\frac{1}{\theta}\sum x_i]$
  - $l(\theta) = -mlog\theta - \frac{1}{\theta}\sum x_i$
  - Score equation, $l'(\theta) = -\frac{m}{\theta} + \frac{\sum x_i}{\theta^2} = 0$. This is the equation that we want to solve.
  - $l''(\theta) = \frac{m}{\theta^2} - \frac{2*\sum x_i}{\theta^3}$
- The iterative formula can be written as

$$\theta_{n+1} = \theta_n - \frac{l'(\theta_n)}{l''(\theta_n)}$$

- Let's implement this in R

```r
# Let's generate 30 random numbers from Exp(theta=2) distribution.
set.seed(261)
x=rexp(30,rate=1/2)
m=length(x)

# We all know that the solution of the loglikelihood is the sample mean
mean(x)
```

```
## [1] 2.370094
```

```r
# Let's see if N-R can give us this solution
# Since we will evaluate over and over again let's define two functions

# Score
l_p_theta=function(t){-m/t+sum(x)/(t^2)}

# double differentiation
l_pp_theta=function(t){m/(t^2)-2*sum(x)/(t^3)}

t_old=0.5 # a reasonable initial guess
iter=1    # to keep track how many times we are updating the equaiton
dif= 1   # can put any number (>0.00001)

while(dif>0.00001){
  t_new=t_old - l_p_theta(t_old)/l_pp_theta(t_old)
  print(paste("iteration: ", iter))
  print(t_new)
  #print(c(t_old, l_p_theta(t_old), l_pp_theta(t_old), t_new))
  dif=abs(t_new-t_old)
  t_old=t_new
  iter=iter+1
}
```

```
## [1] "iteration:  1"
## [1] 0.7205202
## [1] "iteration:  2"
## [1] 1.016204
## [1] "iteration:  3"
## [1] 1.385655
## [1] "iteration:  4"
## [1] 1.792296
```

```
## [1] "iteration:  5"
## [1] 2.143593
## [1] "iteration:  6"
## [1] 2.330579
## [1] "iteration:  7"
## [1] 2.368798
## [1] "iteration:  8"
## [1] 2.370093
## [1] "iteration:  9"
## [1] 2.370094
```

- Let's try a different initial value

```r
t_old=3.5 # a reasonable initial guess
iter=1    # to keep track how many times we are updating the equaiton
dif= 1   # can put any number (>0.00001)
dif
```

```
## [1] 1
```

```r
while(dif>0.00001){
  t_new=t_old - l_p_theta(t_old)/l_pp_theta(t_old)
  print(paste("iteration: ", iter))
  print(t_new)
  #print(c(t_old, l_p_theta(t_old), l_pp_theta(t_old), t_new))
  dif=abs(t_new-t_old)
  t_old=t_new
  iter=iter+1
}
```

```
## [1] "iteration:  1"
## [1] 0.3112324
## [1] "iteration:  2"
## [1] 0.4559131
## [1] "iteration:  3"
## [1] 0.6596116
## [1] "iteration:  4"
## [1] 0.9361054
## [1] "iteration:  5"
## [1] 1.28898
## [1] "iteration:  6"
## [1] 1.692762
## [1] "iteration:  7"
## [1] 2.069001
```

```
## [1] "iteration:  8"
## [1] 2.302216
## [1] "iteration:  9"
## [1] 2.366314
## [1] "iteration:  10"
## [1] 2.370082
## [1] "iteration:  11"
## [1] 2.370094
## [1] "iteration:  12"
## [1] 2.370094
```

- In both cases we found the solution.

Problem with N-R

- N-R is known to suffer heavily if the initial guess is "too far" away from the solution and under some other technical scenarios.
- Let me give you a demonstration

```
t_old=5  # Though I didn't expect 5 to be too far from the solution
iter=1
dif= 1
dif
```

```
## [1] 1
```

```
while(dif>0.00001){
  t_new=t_old - l_p_theta(t_old)/l_pp_theta(t_old)
  print(paste("iteration: ", iter))
  print(c(t_old, l_p_theta(t_old), l_pp_theta(t_old), t_new))
  dif=abs(t_new-t_old)
  t_old=t_new
  iter=iter+1
  if(dif>200){break}
}
```

```
## [1] "iteration:  1"
## [1]  5.00000000 -3.15588724  0.06235489 55.61170066
## [1] "iteration:  2"
## [1]  55.61170066  -0.51646400   0.00887355 113.81433974
## [1] "iteration:  3"
## [1] 113.814339743  -0.258098177   0.002219485 230.101773945
## [1] "iteration:  4"
## [1]  2.301018e+02 -1.290342e-01  5.549339e-04  4.626235e+02
```

- If you look at the value of the second derivative you will see how tiny it is getting which destabilizes the algorithm.

Fisher's Scoring method

- Continuing with the iterative formula given at the bottom of page 10 of this document,

$$\theta_{n+1} = \theta_n - \frac{l'(\theta_n)}{l''(\theta_n)}$$

- This *Fisher's scoring algorithm* is a different version of N-R
  - Instead of using $l''(\theta)$ we use $E[l''(\theta)]$
- So the iterative formula becomes

$$\theta_{n+1} = \theta_n - \frac{l'(\theta_n)}{E[l''(\theta)]|_{\theta=\theta_n}}$$

- In our exponential example, $E[l''(\theta)] = -\frac{m}{\theta^2}$
- Let's implement this

```
# Score
l_p_theta=function(t){-m/t+sum(x)/(t^2)}

# Expected double differentiation
e_l_pp_theta=function(t){-m/t^2}

t_old=5
iter=1
dif= 1

while(dif>0.00001){
  t_new=t_old - l_p_theta(t_old)/e_l_pp_theta(t_old)
  print(paste("iteration: ", iter))
  print(c(t_old, l_p_theta(t_old), e_l_pp_theta(t_old), t_new))
  dif=abs(t_new-t_old)
  t_old=t_new
  iter=iter+1
}
```

```
## [1] "iteration:  1"
## [1]  5.000000 -3.155887 -1.200000  2.370094
## [1] "iteration:  2"
## [1]  2.370094e+00  1.776357e-15 -5.340601e+00  2.370094e+00
```

- We did find the solution and with less number of iterations.
- You will learn these teachniques in details in STA410 (https://fas.calendar.utoronto.ca/course/sta410h1)

Homework

For regression with categorical X

Take the data given in Exercise 10.4.3 (E&R) and do a t-test(assuming population variances are equal). Now fit a model with one dummy variable and compared your numbers from the regression to the numbers from t-test. (If you have time do it by hand or code it yourself without using the functions t.test() or lm())

Repeat the above exercise for the data given in exercise 10.4.6 (E&R)

For Optimization

Try any example (or all of them) that you have done so far in this course where you maximized a likelihood by solving the score equation.