Summer STAT 4206/5206 Final Exam

Christine Chong cc4190

The STAT Summer 4206/5206 final exam is open notes, open book(s), and online resources are allowed. Students are **not** allowed to communicate with any other people regarding this final with the exception of the GU4204 instructor. This includes tutors, friends, classmates and many more… The exam is due through Canvas by Thursday, June 29th, at 11:59pm. Please turn in the .pdf and .Rmd files on Canvas (or .html if you must). Late exams will not be accepted. Good luck!

Goals

The STAT S4206/5206 take-home final exam includes two main components: **Part (1)** Students are required to estimate a logistic regression model via maximum likelihood using Newton's method to optimize the objective log-likelihood function. **Part (2)** Students are required to bootstrap the estimated coefficients to then answer research questions regarding the **prostate** dataset. A description of the dataset and logistic statistical model follow below:

Data Description

A university medical center urology group was interested in the association between prostate-specific antigen (PSA) and a number of prognostic clinical measurements in men with advanced prostate cancer. Data were collected on 97 men who were about to undergo radical prostatectomies. The 8 variables are:

Variable	Variable Name	Description
X_1	PSA level	Serum prostate-specific antigen level (mg/ml)
X_2	Cancer volume	Estimate of prostate cancer volume (cc)
X_3	Weight	Prostate weight (gm)
X_4	Age	Age of patient (years)
X_5	Benign prostatic hyperplasia	Amount of benign prostatic hyperplasia (cm2)
X_6	Seminal vesicle invasion	Presence or absence of seminal vesicle invasion
X_7	Capsular penetration	Degree of capsular penetration (cm)
Y	Gleason score	Pathologically determined grade of disease

Below we read in the dataset and name it **prostate**.

```
prostate <- read.table("FinalExam.txt")
head(prostate)
```

```
##      X1     X2     X3 X4 X5 X6 X7 Y
## 1 0.651 0.5599 15.959 50  0  0  0 6
## 2 0.852 0.3716 27.660 58  0  0  0 7
## 3 0.852 0.6005 14.732 74  0  0  0 7
## 4 0.852 0.3012 26.576 58  0  0  0 6
## 5 1.448 2.1170 30.877 62  0  0  0 6
## 6 2.160 0.3499 25.280 50  0  0  0 6
```

In our setting we create a new binary response variable Y, called high-grade cancer by letting $Y = 1$ if Gleason score equals 8, and $Y = 0$ otherwise (i.e., if Gleason score equals 6 or 7). The goal is to carry out a logistic regression analysis, where the response of interest is high-grade cancer (Y).

```
prostate$Y <- ifelse(prostate$Y==8,1,0)
head(prostate)
```

```
##      X1     X2     X3 X4 X5 X6 X7 Y
## 1 0.651 0.5599 15.959 50  0  0  0 0
## 2 0.852 0.3716 27.660 58  0  0  0 0
## 3 0.852 0.6005 14.732 74  0  0  0 0
## 4 0.852 0.3012 26.576 58  0  0  0 0
## 5 1.448 2.1170 30.877 62  0  0  0 0
## 6 2.160 0.3499 25.280 50  0  0  0 0
```

```
nrow(prostate)
```

```
## [1] 97
```

Logistic Model

Let $Y_1, Y_2, \ldots, Y_{97}$ be independent Bernoulli random variables with expected values

$$E[Y_i] = p_i = \frac{\exp(\beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i} + \beta_4 X_{4i} + \beta_5 X_{5i} + \beta_6 X_{6i} + \beta_7 X_{7i})}{1 + \exp(\beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i} + \beta_4 X_{4i} + \beta_5 X_{5i} + \beta_6 X_{6i} + \beta_7 X_{7i})}, \quad i = 1, 2, \ldots, 97. \tag{1}$$

Part 1: Maximum Likelihood Estimation and Newton's Method

In Model (1), the response values represent high-grade cancer and the features $X_1, X_2, \ldots, X_7$ are outlined in the data description from earlier in this document. To estimate Model (1), we use the method of *Maximum Likelihood*. The objective function of interest (log-likelihood) is

$$\begin{aligned}\ell(\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7) = &\sum_{i=1}^{n} y_i \big(\beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i} + \beta_4 X_{4i} + \beta_5 X_{5i} + \beta_6 X_{6i} + \beta_7 X_{7i}\big) \\ &- \sum_{i=1}^{n} \log\left(1 + \exp(\beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i} + \beta_4 X_{4i} + \beta_5 X_{5i} + \beta_6 X_{6i} + \beta_7 X_{7i})\right)\end{aligned}$$

The above log-likelihood is the same function derived in class except we have several more parameters to estimate, i.e., $\beta_0, \beta_1, \ldots, \beta_7$. In class we only considered *simple logistic regression* (one feature).

Problem 1)

Create a function in **R** called **logistic.NLL** with inputs **beta** and **data**, where **beta** is a vector of β coefficients and **data** is a dataframe defaulted by **data=prostate**. The function **logistic.NLL** should output the negative of the log-likelihood, i.e.,

$$-\ell(\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7).$$

Recall that maximizing the log-likelihood is equivalent to minimizing the negative log-likelihood. Also evaluate the function using the vector **beta=rep(0,8)**, i.e., run the code **logistic.NLL(beta=rep(0,8))**.

```
logistic.NLL <- function(b,data=prostate) {

  b0 <- b[1]
  b1 <- b[2]
  b2 <- b[3]
  b3 <- b[4]
  b4 <- b[5]
  b5 <- b[6]
  b6 <- b[7]
  b7 <- b[8]
  x1 <- data$X1
  x2 <- data$X2
  x3 <- data$X3
  x4 <- data$X4
  x5 <- data$X5
  x6 <- data$X6
  x7 <- data$X7
  y <- data$Y
  #The corresponding probalities that the x matches y
  p.i <- exp(b0+b1*x1+b2*x2+b3*x3+b4*x4+b5*x5+b6*x6+b7*x7)/(1+exp(b0+b1*x1+b2*x2+b3*x3+b4*x4+b5*x5+b6*x
  #Use Negative so we can minimizing instead of maximizing
  return(-sum(dbinom(y,size=1,prob=p.i,log=TRUE)))

}
logistic.NLL(b=rep(0,8))
```

```
## [1] 67.23528
```

Problem 2)

Write a **R** function called **Newtons.Method** that performs *Newton's Optimization Method* on a generic function **f**. The function should have inputs **f**, **x0**, **max.iter**, **stopping.deriv** and **...**. The input **f** is the generic function we wish to minimize, **x0** is the starting point in the Newton's Method algorithm, **max.iter** is the maximum number of iterations (defaulted at 200), **stopping.deriv** is the gradient's threshold at which the algorithm terminates (defaulted at 0.001) and **...** allows you to pass additional arguments, based on **f**, into the **Newtons.Method** function. The output of **Newtons.Method** should be a list giving all updates of our minimizer and the number of iterations required to perform the procedure. You are welcome to add additional outputs if you would like. Hint: this was solved in class.

```
library(numDeriv)

Newton.Method <- function(f, x0, max.iter = 200, stopping.deriv = 0.01, ...){
  n    <- length(x0)
  xmat <- matrix(0, nrow = n, ncol = max.iter)
  xmat[,1] <- x0

  for (k in 2:max.iter) {
    # Calculate the gradient
    grad.cur <- grad(f, xmat[ ,k-1], ...)

    #Calculate the hessian
```

```
    hess.cur <- hessian(f, xmat[ ,k-1], ... )

    # Should we stop?
    if (all(abs(grad.cur) < stopping.deriv)) {
      k <- k-1; break
    }

    # Move in the opposite direction of the grad
    #Use Inverse of hess.cur and matrix multiply with grad curve
    xmat[ ,k] <- xmat[ ,k-1] - solve(hess.cur)%*%grad.cur
  }

  xmat <- xmat[ ,1:k] # Trim
  return(list(x = xmat[,k],
              xmat = xmat,
              k = k,
              minimum=f(xmat[,k],...)
  )
  )
}
```

Problem 3)

Run the function **Newtons.Method** to minimize the function **logistic.NLL** using initial value **x0=rep(0,8)**, maximum iterations **max.iter=200** and **stopping.deriv=.001**. Display the estimated parameters for $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7$. How many iterations did the algorithm take to converge?

The betas are -10.2766092, 0.055228, 0.0868686, 0.0016149, 0.1066326, -0.0718275, -1.1541371, 0.1233242. The algorithm took 7 iterations to converge

Problem 4)

Check that the parameter estimates from the logistic model are reasonably close to the estimates coming from the **glm()** function. Note: The **glm()** code is explicity given below.

```
model <- glm(Y~X1+X2+X3+X4+X5+X6+X7,data=prostate,family=binomial(link = "logit"))
model$coefficients
```

```
##   (Intercept)            X1            X2            X3            X4
## -10.276609553   0.055228000   0.086868614   0.001614891   0.106632576
##            X5            X6            X7
##  -0.071827481  -1.154137232   0.123324157
```

The estimates from the Newton's Method and the glm function are very close to each other up to 4 decimal places.

Problem 5)

Consider a respondent with the following characteristics:

Variable	Variable Name	Actual value
X_1	PSA level	21.3 (mg/ml)
X_2	Cancer volume	8.4 (cc)
X_3	Weight	48.4 (gm)
X_4	Age	68 (years)
X_5	Benign prostatic hyperplasia	4.7 (cm2)
X_6	Seminal vesicle invasion	0
X_7	Capsular penetration	3.2 (cm)

Estimate the probability that this respondent is among the high-grade cancer group. Based on this estimated probability, do you believe that a respondent with these characteristics belongs to the high-grade cancer group?

```
linear.pred <- predict(model,newdata=data.frame(X1=21.3, X2= 8.4, X3= 48.4, X4 = 68, X5= 4.7, X6 = 0, X
probs <- exp(linear.pred)/(1+exp(linear.pred))
probs
```

```
##         1
## 0.2720329
```

Based on the probability I do not think that this respondent is likely to belong to the high-grade cancer group.

Part 2: Research Question Using Bootstrap Methods

For Part 2, we wish to investigate which variables have a higher impact on the likelihood of belonging to the high-grade cancer group. To answer this question, we will bootstrap each estimated coefficient and construct 95% bootstrap intervals. If zero does not fall in the bootstrap interval, then the corresponding feature is a statistically significant variable, and hence does contribute to the odds (or likelihood) of belonging to the high-grade cancer group.

Before coding up the exercise, first define a matrix of indices that will be used in the bootstrap procedure. The code follows below:

```
set.seed(5)
B <- 1000
n <- nrow(prostate)
sample.matrix <- matrix(0,nrow=n,ncol=B)
for (b in 1:B) {

  sample.matrix[,b] <- sample(1:n,replace=T)

}
```

Problem 6)

Write function called **boot.coef** that has two inputs, **sample.index** and **data**. The argument **sample.index** is a vector of length $n = 97$ which represents the bootstrap indices. The second argument **data** (with default **data=prostate**) is the original dataset that will be used in the bootstrap procedure. The output of the

function should yield the estimated parameters for a single bootstrap iteration. Test your function using the code: **boot.coef(sample(1:n,n,replace=TRUE),data=prostate)**.

Note: when constructing **boot.coef** you are welcome to use **Newtons.Method()** to estimate the coefficients. You will also receive full credit if you extract the coefficients using the built in **R** function **glm()**. The function **Newtons.Method()** may cause some numerical instability when completing Problem 7.

```
n = 97
boot.coef<- function(sample.index, data=prostate){
   resampled_rows <- sample.index
   resampled_data <- prostate[resampled_rows, ]
   Y <- resampled_data$Y
   X1 <- resampled_data$X1
   X2 <- resampled_data$X2
   X3 <- resampled_data$X3
   X4 <- resampled_data$X4
   X5 <- resampled_data$X5
   X6 <- resampled_data$X6
   X7 <- resampled_data$X7
   model <- glm(Y~X1+X2+X3+X4+X5+X6+X7,data=resampled_data,family=binomial(link = "logit"))

  return(model$coefficients)
}
boot.coef(sample(1:n,n,replace=TRUE),data=prostate)
```

```
##   (Intercept)            X1            X2            X3            X4
## -17.267454770   0.041282323   0.183498452   0.003049049   0.221157581
##            X5            X6            X7
##  -0.224712456  -2.015092582   0.050182607
```

Problem 7)

Bootstrap the estimated coefficients using some function from the **apply** family (or **plyr**), using the matrix **sample.matrix** and using the function **boot.coef** from Problem 6. Display the standard deviations of each bootstrapped coefficient. These standard deviations represent approximate standard errors.

```
apply.boot<- apply(sample.matrix,2,boot.coef)
sd(apply.boot["X1",])
```

```
## [1] 1.196407
```

```
sd(apply.boot["X2",])
```

```
## [1] 1.73162
```

```
sd(apply.boot["X3",])
```

```
## [1] 0.1600062
```

```
sd(apply.boot["X4",])
```

```
## [1] 2.583537
```

```
sd(apply.boot["X5",])
```

```
## [1] 2.6394
```

```
sd(apply.boot["X6",])
```

```
## [1] 69.53457
```

```
sd(apply.boot["X7",])
```

```
## [1] 3.323749
```

Problem 8)

Construct 95% bootstrap intervals for each feature and the intercept. You can use regular bootstrap intervals or you percentile bootstrap intervals (see Lab 3 solutions). Please be clear in which method you choose. Display the estimated coefficients of the logistic model, the approximate standard errors and bootstrap intervals all in a vertical arrangement. Also append a new column describing whether or not the feature is significant. That is, display the collective information in an arrangement similar to the table displayed below:

```
Cl.X1 <- quantile(apply.boot["X1", ], 0.05)
Cl.X2 <- quantile(apply.boot["X2", ], 0.05)
Cl.X3 <- quantile(apply.boot["X3", ], 0.05)
Cl.X4 <- quantile(apply.boot["X4", ], 0.05)
Cl.X5 <- quantile(apply.boot["X5", ], 0.05)
Cl.X6 <- quantile(apply.boot["X6", ], 0.05)
Cl.X7 <- quantile(apply.boot["X7", ], 0.05)
Cl.Y <- quantile(apply.boot["(Intercept)", ], 0.05)
Cu.X1 <- quantile(apply.boot["X1",], 0.095)
Cu.X2 <- quantile(apply.boot["X2",], 0.095)
Cu.X3 <- quantile(apply.boot["X3",], 0.095)
Cu.X4 <- quantile(apply.boot["X4",], 0.095)
Cu.X5 <- quantile(apply.boot["X5",], 0.095)
Cu.X6 <- quantile(apply.boot["X6",], 0.095)
Cu.X7 <- quantile(apply.boot["X7",], 0.095)
Cu.Y <- quantile(apply.boot["(Intercept)",], 0.095)
```

Variable	Coefficients	Approximate SE	95% Boot-CI (LL)	95% Boot-CI (UL)	Significant
Intercept	$\hat{\beta}_0$	$SE(\hat{\beta}_0)$	-27.8099418	-23.0411125	Yes
X_1	$\hat{\beta}_1$	$SE(\hat{\beta}_1)$	0.0235624	0.0317324	Yes
X_2	$\hat{\beta}_2$	$SE(\hat{\beta}_2)$	-0.1390477	-0.0725699	Yes
X_3	$\hat{\beta}_3$	$SE(\hat{\beta}_3)$	-0.0244874	-0.0141565	Yes
X_4	$\hat{\beta}_4$	$SE(\hat{\beta}_4)$	-0.0020071	0.0217989	No
X_5	$\hat{\beta}_5$	$SE(\hat{\beta}_5)$	-0.546103	-0.3896312	Yes
X_6	$\hat{\beta}_6$	$SE(\hat{\beta}_6)$	-6.7555208	-4.6490869	Yes
X_7	$\hat{\beta}_7$	$SE(\hat{\beta}_7)$	-0.1602073	-0.0897021	Yes

Problem 9)

Write one or two paragraphs to summarize the this dataset. Namely, is there a statistical association between prostate-specific antigen (PSA) and the likelihood of belonging to the high-grade cancer group? Are any other variables statistically impacting the reponse? Also comment on how well the bootstrap procedure performed in this analysis.

According to the average coefficient for X1 from the bootstrap dataset (0.16525) , there is low statistical association between PSA and the likelihood of belonging to the high-grade cancer group. (This is due to the value being less than 0.5). The average coefficient for X6 from the bootstrap dataset (-6.3127291) does show that there is likely a statistical association between Seminal vesicle invasion and the likelihood of belonging to a high-grade cancer group. I think the bootstrap model, while not completely accurate, could be considered good enough to be able to analyze it. I think that if there was more initial data, the model would perform even better because there would be more data to bootstrap.

```
summary(model)
```

```
##
## Call:
## glm(formula = Y ~ X1 + X2 + X3 + X4 + X5 + X6 + X7, family = binomial(link = "logit"),
##     data = prostate)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2381  -0.4578  -0.3430  -0.1310   2.5187
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.276610   4.086002  -2.515   0.0119 *
## X1            0.055228   0.026437   2.089   0.0367 *
## X2            0.086869   0.066132   1.314   0.1890
## X3            0.001615   0.008555   0.189   0.8503
## X4            0.106633   0.061463   1.735   0.0828 .
## X5           -0.071827   0.119903  -0.599   0.5491
## X6           -1.154137   1.061870  -1.087   0.2771
## X7            0.123324   0.113759   1.084   0.2783
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 101.353  on 96  degrees of freedom
## Residual deviance:  61.571  on 89  degrees of freedom
## AIC: 77.571
##
## Number of Fisher Scoring iterations: 6
```