Exam1

Math 534 Exam 1 (in-class)

Mori Jamshidian, March 5, 2024

What to Submit

You must provide your solutions in a single pdf file that includes your R codes. If you use Rmarkdown, submit your .Rmd file in addition to the pdf file. If you are not using Rmarkdown, submit your R codes in a separate text file, but also include the code and output in your pdf file.

This exam has a total of 45 points.

Exam Rules

(a) No form of communication with others is allowed during the exam, except with the instructor.

(b) There are problems that have an astricks (*) next to their point value. I can give you solutions for them, but it will cost you points.

(c) You can use the internet, but not for communications with others or positing questions in any form to any website.

(d) Solutions copied from Chat-GPT , other AIs, or any external source will be considered cheating, and will receive a score of zero.

(e) If you are taking the exam remotely, use Zoom's private chat to communicate with me.

(f) If you are taking the exam remotely, you must always have your camera and speakers on during the exam so that I can see you, and if I give instructions, you can hear me. However, you should mute your microphone unless you need to say something.

(g) I reserve the right to ask for access to your computer during the exam. If I request access, access should be granted immediately. Any delay in access would be considered cheating.

(h) After the exam is graded, you may be asked to explain your solutions in a one-on-one meeting.

(i) The questions on this exam are copyrighted, and should not be distributed or posted in any form at anytime without my consent.

(j) The Rmarkdown file for this exam (`Exam 1.Rmd`) is included. You may write your solutions after each problem in this file, or use your own Rmarkdown file. If you use your own Rmarkdown file, make sure to mark problem numbers clearly, and present the solutions in the order of the problems that appear.

(k) If you are going to take the exam in-person, you need to bring a laptop computer with you.

AI usage policy:

– Chat-GPT and AI are only allowed for simple coding questions, such as "how do I print" a vector in R and the like.

– DO NOT put questions in Chat-GPT or any other AI software:

a) If I realize that any part of your answer comes from AI software, you get a grade of zero for the exam regardless of how you did on the remaining parts of the exam.

b) I have a software that is very good at detecting whether a solution comes from Chat-GPT and a few other AI tools.

c) I often check possible Chat-GPT solutions before I grade. It's easy to catch a Chat-GPT answer, especially when it is wrong!

Good luck with the exam!

Problem 1

Let X be an $n \times p$ matrix of covariates with its i-th row denoted by $\boldsymbol{x}_i^T$, and $\boldsymbol{y}$ be an $n \times 1$ vector of responses with its i-th element y_i. Assume that the distribution of y_i depends on $\boldsymbol{x}_i$ and a set of parameters $\boldsymbol{\beta}$. Specifically, consider the log-likelihood function:

$$\ell(\boldsymbol{\beta}) = C - \sum_{i=1}^{n} \exp\{\boldsymbol{\beta}^T \boldsymbol{x}_i\} + \boldsymbol{\beta}^T \sum_{i=1}^{n} y_i \boldsymbol{x}_i,$$

where C is a constant (which we will ignore from this point on), $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_p)^T$ is a column vector with p elements, and $\boldsymbol{x}_i = (x_{i1}, \cdots, x_{ip})^T$ is a column vector with p elements.

(a)[3* Points] Obtain the first differential $d\ell(d\boldsymbol{\beta})$.

$$d\ell(d\boldsymbol{\beta}) = -\sum_{i=1}^{n} \left\{ \left(d\boldsymbol{\beta}^T \boldsymbol{x}_i \right) \exp(\boldsymbol{\beta}^T \boldsymbol{x}_i) \right\} + d\boldsymbol{\beta}^T \sum_{i=1}^{n} (y_i \boldsymbol{x}_i)$$

(b)[3* Points] Obtain a formula for $\frac{\partial(\ell(\boldsymbol{\beta}))}{\partial \beta_j}$, the partial derivative of $\ell(\boldsymbol{\beta})$ with respect to β_j, where $j = 1, \cdots, p$. Write only your final solution in simplified form.

$$-\sum_{i=1}^{n} [(\boldsymbol{x}_i) \exp(\boldsymbol{\beta}^T \boldsymbol{x}_i)]_j + [\sum_{i=1}^{n} (y_i \boldsymbol{x}_i)]_j$$

(c)[3* Points] The second differential of $\ell(\boldsymbol{\beta})$ is given by

$$ddl(d\boldsymbol{\beta}, d\boldsymbol{\beta}) = -\sum_{i=1}^{n} \left\{ \left(d\boldsymbol{\beta}^T \boldsymbol{x}_i \right) \left(d\boldsymbol{\beta}^T \boldsymbol{x}_i \right) \exp(\boldsymbol{\beta}^T \boldsymbol{x}_i) \right\}.$$

Using this second differential, obtain a formula for $\frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial \beta_j \partial \beta_k}$, the second partial derivative of $\ell(\boldsymbol{\beta})$ with respect to β_j and β_k, where $j, k = 1, \cdots, p$. Write only you final solution in simplified form.

$$-\sum_{i=1}^{n} [x_i]_j [x_i]_k [exp\{\boldsymbol{\beta} \boldsymbol{x}_i\}]_{jk}$$

(d)[3* Points] Write the (j, k)-th element of the information matrix.

Problem 2

The dataset commodity.csv contains daily data on the number of a commodity sold (*sales*), the price of the commodity (*price*), and whether a TV ad was run for the product (*ad*). For the variable *ad*, 0 indicates no TV ad, and 1 indicates TV ad was run. In modeling the number of the commodities sold as a function of the price and the TV ad, we obtain the log-likelihood function given in **Problem 1**, where

- y_i is the number of the commodities sold on day i (given in variable *sales*)
- $\boldsymbol{x}_i$ consists of an intercept term (a 1 in its first element), the price for day i as its second elements, and the TV ad indicator as its third element.
- $\boldsymbol{\beta}$ is a vector pf parameters consisting of β_0, β_1, and β_2, representing the intercept, and the effect of the price, and the effect of the TV ad, respectively.

(a)[3* Points] The function like(y, X, beta, grad = FALSE) given in the file Problem1.R computes the log-likelihood function. Read the commodity.csv file into R, and use the like() function to obtain the value of the log-likelihood at $\beta = (0.1, 0.1, 0.1)$. Show your R code and print the value of the log-likelihood.[Hint: Form a data frame X with the first column being a vector of 1's, the second column being the price, and the third column being the TV ad indicator. Then, use the like() function. It's important that X be a data frame otherwise the code will not work. Also use the c() function in R to specify $\boldsymbol{\beta}$.]

```
comm <- read.csv("commodity.csv", head = T) #reading data
attach(comm)

# likelihood function
like <- function(y, X, beta, grad = FALSE) {
  # X is an n x p data frame consisting of predictors. If there is an intercept,
  #    the first column of X is 1
  # y is an n x 1 vector of responses
  # beta is a p x 1 vector of coefficients
  g = NULL
  ell = NULL
  tmp <- dim(X)
  n <- tmp[1]
  p <- tmp[2]
  v = matrix(0, p, 1)
  ebx <- 0
  ebxx <- matrix(0, p, 1)
  for (i in 1:n) {
    v = v + t(X[i,]*y[i])
    e_xbeta <- exp(as.numeric(as.matrix(X[i,]) %*% beta))
    ebx <- ebx + e_xbeta
  }
  ell <- -ebx + t(v) %*% beta
  list(ell = ell, g = g)
}

# intial beta
beta_init <- c(0.1,.1,.1)
X <- as.data.frame(cbind(1,price,ad))
y <- sales


like(y, X, c(0.1,.1,.1), grad = FALSE)
```

```
## $ell
##        [,1]
## 1 984.9729
##
## $g
## NULL
```

log-like value at β = (0.1,0.1,0.1) is 984.9729

(b)[5* Points] The function like() in the file Problem1.R has an argument grad that is set to FALSE by default. When grad is set to TRUE, the function should compute the gradient of the log-likelihood. Modify the function like() to include the gradient computation. In your R code use the letter g for the gradient. Note that the function outputs ell and g, the values of the log-likelihood and the gradient. Use your modified like() function to obtain the value of the gradient at $\beta = (0.1, 0.1, 0.1)$. Show your R code and print the value of the gradient that you compute.

```r
# likelihood function
like <- function(y, X, beta, grad = FALSE) {
  # X is an n x p data frame consisting of predictors. If there is an intercept,
  #     the first column of X is 1
  # y is an n x 1 vector of responses
  # beta is a p x 1 vector of coefficients
  g = NULL

  ell = NULL
  tmp <- dim(X)
  n <- tmp[1]
  p <- tmp[2]
  v = matrix(0, p, 1)
  ebx <- 0
  ebxx <- matrix(0, p, 1)
  for (i in 1:n) {
    v = v + t(X[i,]*y[i])
    e_xbeta <- exp(as.numeric(as.matrix(X[i,]) %*% beta))
    ebx <- ebx + e_xbeta
  }
  ell <- -ebx + t(v) %*% beta

  # gen gradient
  # component 1
  g0 <- -ebx + sum(sales)
  # component 2
  for (i in 1:n) {
    e_xbeta <- X[i,2]*exp(as.numeric(as.matrix(X[i,]) %*% beta))
    ebx <- ebx + e_xbeta
  }
  g1 <- -ebx + sum(X[,2]*y)
    # component 3
  for (i in 1:n) {
    e_xbeta <- X[i,3]*exp(as.numeric(as.matrix(X[i,]) %*% beta))
    ebx <- ebx + e_xbeta
  }
  g2 <- -ebx + sum(X[,3]*y)

  g = as.matrix(cbind(g0,g1,g2),ncol=1,nrow=3)
  list(ell = ell, g = t(g))
}

like(y, X, c(0.1,.1,.1), grad = T)
```

```
## $ell
##        [,1]
## 1 984.9729
##
## $g
##            [,1]
## g0    648.3959
## g1   4338.4145
## g2 -17960.5403
```

(c)[3 Points] In the context of this problem, is the direction $\mathbf{d} = (1, -1, 1)^T$ an ascent direction at $\boldsymbol{\beta} = (1, 1, 1)$? Why or why not? Show any computations that you use to justify your answer.

```
d = c(1,1,-1)
grad1 <- like(y, X, c(1,1,1), grad = T)$g
d %*% grad1
```

```
##           [,1]
## [1,] -44348587
```

it is descendant in that direction **d**.

Problem 3

(a)[3 Points] Continuing with the likelihood function of Problem 1, the file problem3a.R includes a code for $\nabla^2 \ell(\boldsymbol{\beta})$, the Hessian of the log-likelihood function plus a function called newton() that implements the Newton algorithm. Add the function like() that you completed in Problem 2(a) to the file problem3a.R to be able to compute the log-likelihood and the gradient. The function newton() has two parameters called tol_err and tol_grad that are to be used to control convergence based on *modified relative error* (MRE), and the *norm of the gradient*, respectively. Add these convergence criteria to the newton() function at the location in the file where it says "Add your code for checking convergence here." Once done, set both tol_err and tol_grad to 10^{-9}, and use the newton() function to obtain the MLE of $\boldsymbol{\beta}$ starting at $\boldsymbol{\beta} = (0.1, 0.1, 0.1)$. Show your R code and print the MLE of $\boldsymbol{\beta}$. Make sure to keep the print statement in the newton() function that shows the iterative process.

```
# likelihood function
like <- function(y, X, beta, grad = FALSE) {
  # X is an n x p data frame consisting of predictors. If there is an intercept,
  #    the first column of X is 1
  # y is an n x 1 vector of responses
  # beta is a p x 1 vector of coefficients


  g = NULL

  ell = NULL
  tmp <- dim(X)
   n <- tmp[1]
   p <- tmp[2]
  v = matrix(0, p, 1)
  ebx <- 0
  ebxx <- matrix(0, p, 1)
  for (i in 1:n) {
    v = v + t(X[i,]*y[i])
    e_xbeta <- exp(as.numeric(as.matrix(X[i,]) %*% beta))
    ebx <- ebx + e_xbeta
  }
  ell <- -ebx + t(v) %*% beta

  # gen gradient
  # component 1
  g0 <- -ebx + sum(sales)
  # component 2
  for (i in 1:n) {
    e_xbeta <- X[i,2]*exp(as.numeric(as.matrix(X[i,]) %*% beta))
    ebx <- ebx + e_xbeta
  }
  g1 <- -ebx + sum(X[,2]*sales)
    # component 3
  for (i in 1:n) {
    e_xbeta <- X[i,3]*exp(as.numeric(as.matrix(X[i,]) %*% beta))
    ebx <- ebx + e_xbeta
  }
  g2 <- -ebx + sum(X[,3]*sales)

  g = as.matrix(cbind(.6*g0,.6*g1,.6*g2),ncol=1,nrow=3)
  list(ell = ell, g = t(g))
}

Hess <- function(y, X, beta) {
  # X is an n x p matrix of predictors
  # y is an n x 1 vector of responses
  # beta is a p x 1 vector of coefficients

  tmp <- dim(X)
  n <- tmp[1]
  p <- tmp[2]
  H <- matrix(0, p, p)
  for( i in 1:n){
    e_xbeta <- exp(as.matrix(X[i,]) %*% beta)
    for (k in 1:p) {
      for(m in 1:p) {
        H[k,m] <- H[k,m] - e_xbeta * X[i,k] * X[i,m]
      }
    }
  }
  return(H)
}

#create Newton's method for maximization
newton <- function(y, X, beta, maxit = 100, tol_err = 1e-9, tol_grad = 1e-9) {
  # X is an n x p matrix of predictors
  # y is an n x 1 vector of responses
  # beta is a p x 1 vector of coefficients
  # maxit is the maximum number of iterations
  # tol is the convergence toleranc

  # initialize the parameters
  tmp <- dim(X)
  n <- tmp[1]
  p <- tmp[2]
  beta_old <- beta
  beta_new <- beta
  # iterate until convergence
  for (iter in 1:maxit) {
    # compute the gradient and Hessian
    a <- like(y, X, beta_new, grad = TRUE)
    g <- a$g
```

```r
    H <- Hess(y, X, beta_new)
    # update the parameters
    beta_old <- beta_new
    beta_new <- beta_old - solve(H) %*% (g)
    # Add your code for checking convergence here
    mre = max(abs(beta_old - beta_new)/abs(pmax(1,abs(beta_old))))
    if (mre < tol_err & norm(a$g) < tol_grad){break}
    # Keep the print statement below for your final version
    print(c(iter, beta_new, a$ell, norm(a$g, "2"), mre))
  }
  # return the estimated parameters
  return(beta_new)
}

newton(y,X,c(.1,.1,.1))
```

(b)[2 Points] Obtain the standard errors of the MLE of $\boldsymbol{\beta}$ obtained in part (a). Show your R code and print the standard errors.

Problem 4

The file Population.csv consists of data on the U.S. population (in millions) from 1790 to the latest census data in 2020. The logistic model

$$y_i = \frac{\beta_1}{1 + \exp(\beta_2 + \beta_3 t_i)} + \varepsilon_i$$

is a model used for population growth, where y_i is the population size at time t_i, β_1 is the limit towards which the population grows, β_2 is a relative measure of the size population at time 0 as compared to β_3, and β_3 is the growth rate of the population. In this problem, you will use the nls() function in R to fit the logistic model given above to the U.S. population data in the file Population.csv.

Note: In order not to run into numerical computation issues, transform the year to t = (Year - 1790)/10, which means that year 1790 is 0 and year 2020 is 23.

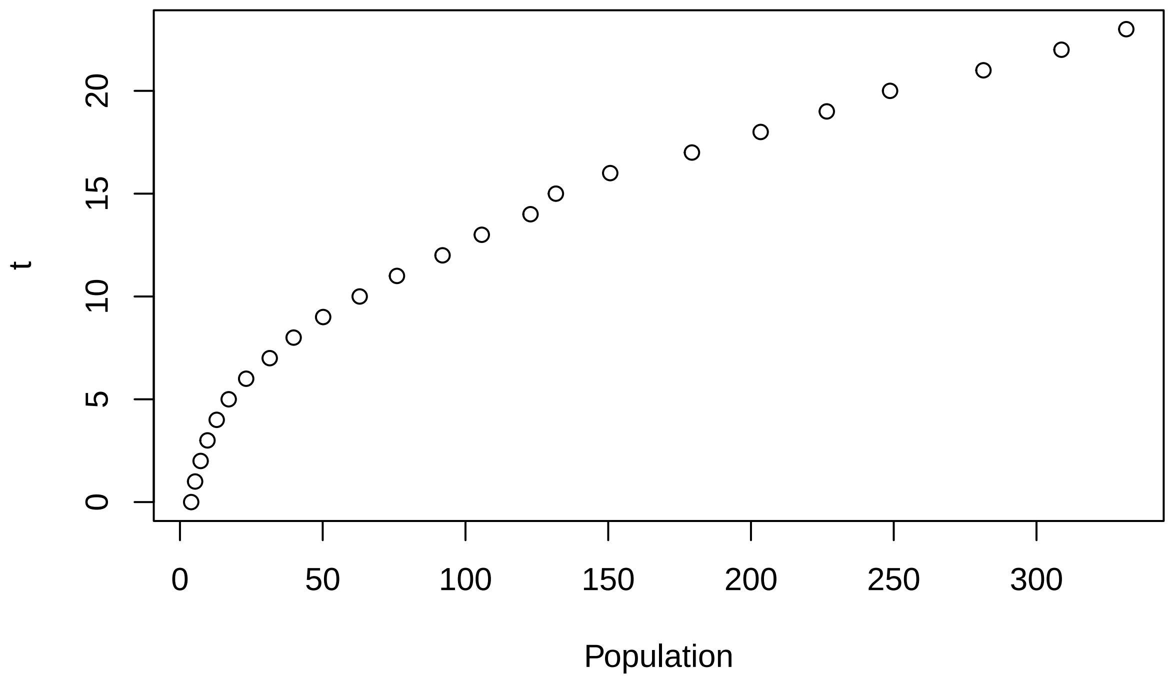
(a)[3* Points] Obtain initial values for the parameters β_1, β_2, and β_3 by linearizing the model, and fitting a linear regression model to the linearized version. [Hint: the value of β_1 is the population in the long run, so you would fix that to a large value, say 500. Then take the reciprocal of y_i, and do some algebraic manipulations to get a linear model in terms of t_i.] You do not need to show your algebraic manipulations. Just type the linear model that you fit, show your R code and print the initial values of the parameters.

```r
pop <- read.csv("Population.csv")
head(pop)
```

```
##   Year Population
## 1 2020     331.45
## 2 2010     308.75
## 3 2000     281.42
## 4 1990     248.71
## 5 1980     226.55
## 6 1970     203.39
```

```r
attach(pop)
t <- (Year - 1790)/10

plot(Population,t)
```



(b)[5* Points] Use R's nls() function to fit the logistic model to the data. Use the following options in your code: trace = TRUE, nls.control(maxiter = 500, tol = 1e-7, minFactor = 1/1024, printEval=TRUE) Include your R code, the iteration output, and final parameter estimates.

The formulas for the derivatives that you need are as follows:

$$f_i = \frac{\beta_1}{1 + \exp(\beta_2 + \beta_3 t_i)}$$
$$\partial f_i / \partial \beta_1 = \frac{1}{1 + \exp(\beta_2 + \beta_3 t_i)}$$
$$\partial f_i / \partial \beta_2 = \frac{-\beta_1 \exp(\beta_2 + \beta_3 t_i)}{(1 + \exp(\beta_2 + \beta_3 t_i))^2}$$
$$\partial f_i / \partial \beta_3 = \frac{-\beta_1 t_i \exp(\beta_2 + \beta_3 t_i)}{(1 + \exp(\beta_2 + \beta_3 t_i))^2}.$$

```r
model <- function(t,b1,b2,b3){
  E = exp(b2 + b3*t)
  f <- b1/(1 + E)
  # grad
  grad <- cbind(1/(1+E),
                -b1*E/(1+E)^2,
                -b1*t*E/(1+E)^2)
  attr(f,'gradient') <- grad
  f
}
t <- (Year - 1790)/10
nls_fit <-  nls(Population ~ model(t,b1,b2,b3),
                start = list(b1=.1,b2=.1,b3=.1),
                trace = T,
                nls.control(maxiter = 500, tol = 1e-7, minFactor = 1/1024,
                            printEval=TRUE))
```

(c)[4 Points] Plot the data and the fitted model on the same axes. What does your model predict for the population in the year 2050? Show your computation.

Problem 5

(a)[3 Points] Consider the data given in Problem 2 on the daily sales of a commodity. Let y_i denote the number of commodities sold on the i-th day, with the corresponding price of x_{i1} and the TV ad indicator of x_{i2}. Moreover, assume that y_i has a Poisson distribution with mean

$$E(y_i) = \exp\{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}\}.$$

What weighted nonlinear least squares needs to be fitted at each iteration of the Iteratively Reweighted Least Squares (IRLS) algorithm to obtain the MLE of $\boldsymbol{\beta}$ (no coding required).Specify the model and the weights. A generic solution will not get any points. Your solution must be specific to the problem described here. [Hint: Variance of a Poisson random variable is equal to its mean].

$$[W]_{ii} = exp\{-\beta_0 - x_{i1}\beta_0 - x_{i2}\beta_2\}$$

the weights used will be an inverse of the variance $exp\{\beta_0 + x_{i1}\beta_0 + x_{i2}\beta_2\}$. This will give us the above W matrix. while the model we use will be $\exp\{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}\} + \epsilon_i$. Things here are simple because this si s a

(b)[2 Points] Do the parameter values $\beta_1 = -5$, $\beta_2 = 3$, and $\beta_3 = -2$ provide a feasible solution? Why or why not?