Homework 4

AUTHOR
Ashutosh Ekade

Question 1: Linear Regression with Normal Errors

Load the `BostonHousing2` dataset from the `mlbench` package, which has data on 506 census tracts from the 1970's. Assume the model is $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + e$ with $e \sim N(0, \sigma^2)$, observations are independent, and where Y denotes `medv` (the median value of homes in 1000's of dollars), X_1 denotes `rm` (the average number of rooms per home), X_2 denotes `age` (the proportion of older homes), and X_3 denotes `crim` (the crime rate in the area). See `?mlbench::BostonHousing2` for definitions of these variables.

```r
#install.packages('mlbench')
library(mlbench)
data('BostonHousing2')
```

1.a) Write, mathematically, the joint log-likelihood function. What link function (or inverse link function) is required to "connect" μ_i to $x_i'\beta$?

$$

$$
\begin{aligned}
l(\beta, \sigma^2) &= \sum_{i=1}^{n} \log f(Y_i | Y_i, \beta, \sigma^2) \\
&= -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i - X_i'\beta)^2 \\
&= -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y - X\beta)'(Y - X\beta)
\end{aligned}
$$

$$

1.b) Write an `R` function to calculate the log-likelihood function from 1a above. Your function should take 3 arguments: (1) $\theta = (\beta, \sigma^2)$ a vector of all (five) parameters, (2) an $n \times k$ matrix X, and (3) a vector or $n \times 1$ matrix y.

```r
log_likelihood <- function(theta, X, y) {
      beta <- theta[1:4]
      sigma_sq <- theta[5]
      # log-likelihood
      n <- length(y)
      residuals <- y - X %*% beta
      log_likelihood_value <- -n/2 * log(2 * pi) - n/2 * log(sigma_sq) - 1/(2 * sigma_
    return(log_likelihood_value)
  }
```

1.c) Use `optim()` and your function from 1b to find $\hat{\beta}_{\text{MLE}}$ and $\hat{\sigma}^2_{\text{MLE}}$ as well as their standard errors. You may find it helpful to initialize your search at 0 for the β parameters and $\text{var}(y)$ for σ^2. You will likely find some differences in your standard errors compared to the output from `lm()` in 1d below. This is because `lm()` uses analytical expressions for the standard errors, whereas `optim()` uses numerical approximations to the Hessian matrix.

```
set.seed(1234)
X <- cbind(1,BostonHousing2$rm,BostonHousing2$age,BostonHousing2$crim)
Y <- matrix(BostonHousing2$medv,ncol=1)
initial_parameters <- c(1,1,1,1,var(Y))
#optimization function is called
result <- optim(par = initial_parameters, fn = log_likelihood,  X=X, y=Y,control = lis
```

```
Warning in log(sigma_sq): NaNs produced

Warning in log(sigma_sq): NaNs produced
```

```
print(result)
```

```
$par
[1] -23.60366982   8.03253919  -0.05224289  -0.21102912  36.83582886

$value
[1] -1630.435

$counts
function gradient 
      58       23 

$convergence
[1] 0

$message
NULL

$hessian
              [,1]          [,2]          [,3]          [,4]          [,5]
[1,] -1.373663e+01 -8.632967e+01 -9.419878e+02 -4.963762e+01 -5.229595e-06
[2,] -8.632967e+01 -5.493184e+02 -5.854902e+03 -2.937887e+02 -8.458301e-05
[3,] -9.419878e+02 -5.854902e+03 -7.545954e+04 -4.574757e+03 -3.916512e-05
[4,] -4.963762e+01 -2.937887e+02 -4.574757e+03 -1.193684e+03 -3.902301e-05
[5,] -5.229595e-06 -8.458301e-05 -3.916512e-05 -3.902301e-05 -1.864794e-01
```

```
#MLE estimates
beta_mle <- result$par[1:4]
sigma_square_mle <- result$par[5]
std_error <- sqrt(-diag(solve(result$hessian)))
print("Beta MLE")
```

```
[1] "Beta MLE"
```

```
print(beta_mle)
```

```
[1] -23.60366982   8.03253919  -0.05224289  -0.21102912
```

```
print("Sigma Square MLE")
```

```
[1] "Sigma Square MLE"
```

```
print(sigma_square_mle)
```

```
[1] 36.83583
```

```
print("Standard Error")
```

```
[1] "Standard Error"
```

```
print(std_error)
```

```
[1] 2.75832954 0.40040222 0.01042100 0.03392967 2.31571187
```

1.d) Use `lm()` and `summary()` to check your work in 1c above.

```
summary(lm(BostonHousing2$medv ~ BostonHousing2$rm+BostonHousing2$age+BostonHousing2$c
```

```

Call:
lm(formula = BostonHousing2$medv ~ BostonHousing2$rm + BostonHousing2$age + 
    BostonHousing2$crim)

Residuals:
    Min      1Q  Median      3Q     Max 
-19.959  -3.143  -0.633   2.150  39.940 

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)        -23.60556    2.76938  -8.524  < 2e-16 ***
BostonHousing2$rm    8.03284    0.40201  19.982  < 2e-16 ***
BostonHousing2$age  -0.05224    0.01046  -4.993 8.21e-07 ***
BostonHousing2$crim -0.21102    0.03407  -6.195 1.22e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.094 on 502 degrees of freedom
Multiple R-squared:  0.5636,	Adjusted R-squared:  0.561 
F-statistic: 216.1 on 3 and 502 DF,  p-value: < 2.2e-16
```

1.e) Do the signs on the 3 estimated slope coefficients make sense? Why or why not?

The estimated slope coefficients align with expectations for the median home value determination. As generally observed, the number of rooms per home shows a positive relationship, while both age and crime rate exhibit inverse correlations with home values. Older houses typically depreciate in value over time, reflecting a negative correlation, while higher crime rates often decrease the desirability of an area, leading to lower demand and subsequently lower home values. Therefore, the estimated coefficients in the model are anticipated to be negative, reflecting the adverse effects of house age and crime rate on the estimated median home value.

1.f) In 1c, `optim()` returned the hessian matrix, which enabled you to calculate the standard errors of the MLEs. In 1d, `lm()` provided the standard errors from analytical expressions we derived in class. Suppose, however, that you were unable to calculate the standard errors and needed to use a bootstrap to estimate them. Write `R` code to perform a bootstrap to estimate the standard errors of the 3 $\hat{\beta}_{\text{MLE}}$ parameters. Each time through the loop, you may use `lm()` and `coef()` if you would like, or you may calculate $\hat{\beta}$ directly from matrices; you may **not** use the `boot()` function from the `boot` package.

```r
set.seed(1234)
B <- 1000
n <- nrow(X)
res <- matrix(NA_real_, nrow=B, ncol=ncol(X))
X_test <- X[,c(2,3,4)]

for(b in 1:B) {
  draws <- sample(1:n, size=n, replace=T)
  Y_boot <- Y[draws,]
  X_boot <- X_test[draws,]
  res[b,] <- lm(Y_boot ~ X_boot)$coef
}

serr <- apply(res, 2, function(x) sqrt(var(x)))

print("The standard errors estimated from the bootstrap are:")
```

```
[1] "The standard errors estimated from the bootstrap are:"
```

```r
serr[cbind(2,3,4)]
```

```
[1] 0.67701162 0.00927441 0.03502999
```

1.g) Use your estimates and standard errors from 1c above to test whether the 3 β slope coefficients are each (separately) statistically significantly different from zero at a 95% confidence level. Check your results against the output in 1d above.

```r
alpha <- 0.05
critical_value <- qt(1-alpha/2,n-3)
beta_stat <- beta_mle/std_error[1:4]
```

```r
print("Beta Statistics")
```

```
[1] "Beta Statistics"
```

```r
beta_stat
```

```
[1] -8.557233 20.061175 -5.013230 -6.219605
```

```r
critical_value <- qt(1 - alpha / 2, n-3)
print("Critical Value")
```

```
[1] "Critical Value"
```

```r
print(critical_value)
```

```
[1] 1.964691
```

```r
print("All the 3 slope coefficients are statistically significant (at 95% confidence l
```

```
[1] "All the 3 slope coefficients are statistically significant (at 95% confidence
level) because absolute value of each t-stat is greater than critical value."
```

Question 2: A Poisson Model for Count Data

Load the `trading_behavior` dataset.

The data provides 200 observations on equity trading behavior of Anderson students. `id` is an anonymized identifier for the student. `numtrades` is the median weekly number trades made by each student during the Fall quarter. `program` indicates whether the student is in the MSBA (1), MBA (2), or MFE (3) program (note that you may need to store this variable as a factor or convert it to a set of dummy variables when using it to fit a statistical model). `finlittest` is the students' scores on a financial literacy test taken before entering their graduate program (higher scores indicate higher financial "literacy").

Assume you want to model the number of trades as a function of graduate program (where \mathbbm1 is an indicator function) and financial literacy:

$$y_i \sim \text{Pois}(\mu_i)$$

$$\log \mu_i = \beta_0 + \beta_1 \text{\mathbbm1}(MBA) + \beta_2 \text{\mathbbm1}(MFE) + \beta_3 \text{finlittest}$$

```r
trading_behavior <- read.csv("trading_behavior.csv")
head(trading_behavior)
```

```
   id numtrades program finlittest
1  45         0     MFE         66
2 108         0    MSBA         66
3  15         0     MFE         69
4  67         0     MFE         67
5 153         0     MFE         65
6  51         0    MSBA         67
```

2.a) A Poisson density for random variable Y with parameter μ is $f(y|\mu) = \exp(-\mu)\mu^y/y!$. Suppose we let each Y_i have it's own parameter μ_i with link function $\log(\cdot)$: specifically, $\log(\mu_i) = x_i'\beta$. Assume the data are sampled independently. Write, mathematically, the joint log-likelihood function.

$$

$$
f(y|\mu) = \frac{e^{-\mu}\mu^y}{y!}
$$
$$
\log(\mu_i) = \beta_0 + \beta_1 \times 1(\text{MBA}) + \beta_2 \times 1(\text{MFE}) + \beta_3 \times \text{finlittest}
$$
$$
\log f(y_i|\mu_i) = \log\left(\frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!}\right) = -\mu_i + y_i\log(\mu_i) - \log(y_i!)
$$
$$
\ell(\beta) = \sum_{i=1}^{n}[-\mu_i + y_i\log(\mu_i) - \log(y_i!)]
$$
$$
\ell(\beta) = \sum_{i=1}^{n}[-\mu_i + y_i\log(\mu_i) - \log(\Gamma(y_i+1))]
$$
$$
\mu_i = e^{(\beta_0+\beta_1\times 1(\text{MBA}_i)+\beta_2\times 1(\text{MFE}_i)+\beta_3\times\text{finlittest}_i)}
$$

$$

2.b) Write an R function to calculate the log-likelihood function from 2a above. Your function should take 3 arguments: (1) a vector β of all (four) parameters, (2) an $n \times k$ matrix X, and (3) a vector or $n \times 1$ matrix y.

```r
trading_behavior$program <- relevel(factor(trading_behavior$program), ref = 'MSBA')

x <- model.matrix(~ program + finlittest, data = trading_behavior)
y <- as.matrix(trading_behavior$numtrades)
initial_beta <- as.vector(rep(0, ncol(x)))

loglik_poisson <- function(beta, x, y) {

    predicted_values <- x %*% beta

    mu <- exp(predicted_values)

    llp <- sum(-mu + y * predicted_values - lgamma(y + 1))

    return(llp)
    }

print(loglik_poisson(initial_beta, x, y))
```

```
[1] -247.6471
```

2.c) Use `optim()` and your function from 1b to find $\hat{\beta}_{\text{MLE}}$ as well as their standard errors.

```r
initial_beta <- as.vector(rep(0, ncol(x)))

set.seed(1234)
result <- optim(par=initial_beta,
            fn=loglik_poisson, x=x,
            y=y, method = "BFGS",
            control=list(fnscale=-1),
            hessian = TRUE)
```

```
beta_mle_poisson <- result$par
std_errors_mle_poisson <- sqrt(-diag(solve(result$hessian)))

print(beta_mle_poisson)
```

```
[1] -6.8385058  1.0859384  0.3604895  0.0682507
```

```
print(std_errors_mle_poisson)
```

```
[1] 0.8917300 0.3560795 0.4389272 0.0105377
```

2.d) Fit the model using `glm()`. Compare your results to the output from `optim` in question 2c above.

```
glm(y ~ x - 1, data=trading_behavior, family = poisson())
```

```
Call:  glm(formula = y ~ x - 1, family = poisson(), data = trading_behavior)

Coefficients:
x(Intercept)   xprogramMBA   xprogramMFE   xfinlittest
    -7.00093       1.08386       0.36981       0.07015

Degrees of Freedom: 200 Total (i.e. Null);  196 Residual
Null Deviance:      319.2
Residual Deviance: 189.4    AIC: 373.5
```

2.e) The "analog" to the F-test from linear regression is the Likelihood Ratio Test. The Likelihood Ratio test statistic is calculated as:

$$LR_n = 2 \times [\ell_n(\hat{\theta}) - \ell_n(\tilde{\theta})]$$

where $\ell_n(\cdot)$ is the log likelihood function, $\hat{\theta}$ is the MLE, and $\tilde{\theta}$ is a constrained parameter vector (e.g., suppose a Null Hypothesis is that $\theta_2 = 0$ & $\theta_3 = 0$). The Likelihood Ratio test statistic LR_n has an asymptotic chi-squared distribution with k degrees of freedom (i.e., χ^2_k where k is the length of the θ vector).

Test the joint hypothesis that $\beta_2 = 0$ & $\beta_3 = 0$ at the 95% confidence level using a Likelihood Ratio test. Specifically, use your log-likelihood function from 2b above and your parameter estimates from 2c above to calculate $\ell_n(\widehat{\beta_{\text{MLE}}})$. Then replace β_2 and β_3 with their hypothesized values and re-calculate the log-likehood (ie, $\ell_n([\hat{\beta}_1, 0, 0, \hat{\beta}_4])$. Next, compute LR_n and compare the value to the cut-off of a chi-squared distribution with 4 degrees of freedom to assess whether or not you reject the Null Hypothesis.

```
loglik_mle <- loglik_poisson(beta_mle_poisson, x, y)
```

```r
beta_constrained <- beta_mle_poisson
beta_constrained[2:3] <- 0

loglik_constrained <- loglik_poisson(beta_constrained, x, y)

# Log likelihood ratio test statistic calculation
LR_t <- 2 * (loglik_mle - loglik_constrained)

# Degrees of freedom = number of constrained parameters
df <- 2

# Critical value from chi-squared distribution
chi_squared_critical <- qchisq(0.95, df)

cat("Likelihood Ratio Test Statistic:", LR_t, "\n")
```

```
Likelihood Ratio Test Statistic: 90.36276
```

```r
cat("Chi-squared critical value at 95% confidence level:", chi_squared_critical, "\n")
```

```
Chi-squared critical value at 95% confidence level: 5.991465
```

```r
# Decision
if (abs(LR_t) > chi_squared_critical) {
  cat("Reject the null hypothesis: beta_2 and beta_3 is non-zero.")
  } else {
  cat("Fail to reject the null hypothesis: beta_2 and beta_3 may both be zero.")
    }
```

```
Reject the null hypothesis: beta_2 and beta_3 is non-zero.
```

We reject the null hypothesis which means that both beta_2 and beta_3 are zero as the test statistic is greater than the chi-squared value at the 95% confidence interval.

Question 3 is OPTIONAL. If you accurately complete it, you will receive 2 bonus points toward your final grade.

Question 3: Estimating Demand via the Multi-Nomial Logit Model (MNL)

Suppose you have $i = 1, \dots, n$ consumers who each select exactly one product j from a set of J products. The outcome variable is the identity of the product chosen $y_i \in \{1, \dots, J\}$ or equivalently a vector of $J-1$ zeros and 1 one, where the 1 indicates the selected product. For example, if the third product was chosen out of 4 products, then either $y = 3$ or $y = (0, 0, 1, 0)$ depending on how you want to represent it. Suppose also that you have a vector of data on each product x_j (eg, size, price, etc.).

The MNL model posits that the probability that consumer i chooses product j is:

$$\mathbb{P}_i(j) = \frac{e^{x_j'\beta}}{\sum_{k=1}^J e^{x_k'\beta}}$$

For example, if there are 4 products, the probability that consumer i chooses product 3 is:

$$\mathbb{P}_i(3) = \frac{e^{x_3'\beta}}{e^{x_1'\beta} + e^{x_2'\beta} + e^{x_3'\beta} + e^{x_4'\beta}}$$

A clever way to write the individual likelihood function for consumer i is the product of the J probabilities, each raised to the power of an indicator variable (δ_{ij}) that indicates the chosen product:

$$L_i(\beta) = \prod_{j=1}^J \mathbb{P}_i(j)^{\delta_{ij}} = \mathbb{P}_i(1)^{\delta_{i1}} \times \ldots \times \mathbb{P}_i(J)^{\delta_{iJ}}$$

Notice that if the consumer selected product $j = 3$, then $\delta_{i3} = 1$ while $\delta_{i1} = \delta_{i2} = \delta_{i4} = 0$ and the likelihood is:

$$L_i(\beta) = \mathbb{P}_i(1)^0 \times \mathbb{P}_i(2)^0 \times \mathbb{P}_i(3)^1 \times \mathbb{P}_i(4)^0 = \mathbb{P}_i(3) = \frac{e^{x_3'\beta}}{\sum_{k=1}^J e^{x_k'\beta}}$$

The joint likelihood (across all consumers) is the product of the n individual likelihoods:

$$L_n(\beta) = \prod_{i=1}^n L_i(\beta) = \prod_{i=1}^n \prod_{j=1}^J \mathbb{P}_i(j)^{\delta_{ij}}$$

And the joint log-likelihood function is:

$$\ell_n(\beta) = \sum_{i=1}^n \sum_{j=1}^J \delta_{ij} \log(\mathbb{P}_i(j))$$

Use the `yogurt_data` dataset, which provides the anonymized consumer identifiers (`id`), a vector indicating the chosen product (`y1`:`y4`), a vector indicating if any products were "featured" in the store as a form of advertising (`f1:f4`), and the products' prices (`p1:p4`). For example, consumer 1 purchased yogurt 4 at a price of 0.079/oz and none of the yogurts were featured/advertised at the time of consumer 1's purchase. Consumers 2 through 7 each bought yogurt 2, etc.

Let the vector of product features include brand dummy variables for yogurts 1-3 (omit a dummy for product 4 to avoid multi-collinearity), a dummy variable to indicate featured, and a continuous variable for price:

$$x'_j = [\text{\mathbbm}1(\text{Yogurt } 1), \text{\mathbbm}1(\text{Yogurt } 2), \text{\mathbbm}1(\text{Yogurt } 3), X_f, X_p]$$

You will need to create the product dummies. The variables for featured and price are included in the dataset. The "hard part" of this likelihood function is organizing the data.

Your task: Code up the log-likelihood function. Use `optim()` to find the MLEs for the 5 parameters ($\beta_1, \beta_2, \beta_3, \beta_f, \beta_p$).

(Hint: you should find 2 positive and 1 negative product intercepts, a small positive coefficient estimate for featured, and a large negative coefficient estimate for price.)

```r
## finished this partly with the help of ChatGPT and partly with my study group.
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.3     ✔ readr     2.1.4
✔ forcats   1.0.0     ✔ stringr   1.5.0
✔ ggplot2   3.4.3     ✔ tibble    3.2.1
✔ lubridate 1.9.2     ✔ tidyr     1.3.0
✔ purrr     1.0.2     
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
yogurt_data <- read.csv("yogurt_data.csv")
yogurt_data <- yogurt_data %>%
  mutate(
    yogurt_1 = as.integer(y1 == 1),
    yogurt_2 = as.integer(y2 == 1),
    yogurt_3 = as.integer(y3 == 1)
  )
log_likelihood <- function(beta, data) {
  x1Beta <- beta[1] + beta[4] * data$f1 + beta[5] * data$p1
  x2Beta <- beta[2] + beta[4] * data$f2 + beta[5] * data$p2
  x3Beta <- beta[3] + beta[4] * data$f3 + beta[5] * data$p3
  x4Beta <- 0 + beta[4] * data$f4 + beta[5] * data$p4
  prob_1 <- exp(x1Beta) / (exp(x1Beta) + exp(x2Beta) + exp(x3Beta) + exp(x4Beta))
  prob_2 <- exp(x2Beta) / (exp(x1Beta) + exp(x2Beta) + exp(x3Beta) + exp(x4Beta))
  prob_3 <- exp(x3Beta) / (exp(x1Beta) + exp(x2Beta) + exp(x3Beta) + exp(x4Beta))
  prob_4 <- exp(x4Beta) / (exp(x1Beta) + exp(x2Beta) + exp(x3Beta) + exp(x4Beta))
  ll <- sum(data$yogurt_1 * log(prob_1) +
            data$yogurt_2 * log(prob_2) +
              data$yogurt_3 * log(prob_3) +
              (1 - data$yogurt_1 - data$yogurt_2 - data$yogurt_3) * log(prob_4))
  return(ll)
}
out_optim <- optim(par = rep(0, 5), fn = log_likelihood, data = yogurt_data, control =
out_optim$par
```

```
[1]   1.3908347   0.6440157  -3.0887416   0.4861072 -37.1591372
```

This matches with the hint provided (2 positive and 1 negative product intercepts, a small positive coefficient estimate for featured, and a large negative coefficient estimate for price.)