Assignment Part I

Silpa Soni Nallacheruvu (19980824-5287), Elva Wallimann (19780306-T063)

2024-09-27

set.seed(900101)

Summary

The Assignment focuses on applying knowledge related to Maximum Likelihood Estimate, Logistic Regression using the Newton-Raphson's Algorithm. The first task is about deriving parameter estimates using score vector and Fisher Information Matrix iteratively until convergence is achieved. The second task is about verifying the accuracy of the NR algorithm against the built-in R's results of the same estimates. The third task is about calculating the standard errors from the MLE and verifying the consistency of the usage of the Fisher Information Matrix. The fourth task is introducing bootstrapping samples to approximate the estimates and constructing confidence interval for a requested use case. We have seen that the NR algorithm and the bootstrap approximations are consistent with the built-in R results.

Task 1

Approach :

We first define functions for the likelihood (L), log-likelihood (l), score function (S), and the Fisher information (I), as specified in the task description. Then we use the defined functions to build a function (NR) for the Newton-Raphson's algorithm to compute the ML-estimates in a logistic regression model.

Code :

```
## # Likelihood
## L <- function(theta, y, X){
##   # Compute the linear predictor
##   eta <- X %*% theta   # dim: N*1
##
##   # Compute the probabilities using the logistic function
##   p <- 1/(1 + exp(-eta))   # dim: N*1
##
##   likelihood <- prod(dbinom(y, size=1, prob=p))  # scalar
##   return(likelihood)
## }
##
##
## # Log-likelihood
## l <- function(theta, y, X){
##   likelihood <- L(theta, y, X)
##   log_likelihood <- log(likelihood)
##   return(log_likelihood)
## }
```

```
##
##
## # Score function
## S <- function(theta, y, X){
##   p <- 1/(1 + exp(-X %*% theta))
##   score <- t(X) %*% (y-p)   # dim: k*1
##   return(as.vector(score))
## }
##
##
## # Fisher information
## I <- function(theta, y, X){
##   p <- 1/(1 + exp(-X %*% theta))
##   v <- p*(1-p)
##   D <- diag(as.vector(v))
##   info <- t(X) %*% D %*% X     # dim: k*k
##   return(info)
## }
##
##
## #---- Task-1 b) Use Newton-Raphson's algorithm to compute the ML-estimates
## # in a logistic regression model ----
##
## NR <- function(theta0, niter, y, X) {
##   # Initialize theta with the starting value
##   theta <- theta0
##
##   for (i in 1:niter) {
##     score <- S(theta, y, X)
##     info <- I(theta, y, X)
##
##     # Update theta using the Newton-Raphson formula
##     theta <- theta + solve(info) %*% score
##
##     # Print current estimate (for tracking)
##     cat("Iteration", i, ": theta =", theta, "\n")
##   }
##
##   return(theta)  # Return the final estimate
## }
```

Task 2

Approach :

We verified our function NR from task 1 by re-computing the parameter estimates provided by R (R actually uses the same type of algorithm). We first imported data from https://raw.githubusercontent.com/mskold SU/MT5003_HT17/master/Projekt/proj_data.csv and set random seed to 900101 to sample 1000 entries. Then, we estimate a logistic regression model using the embeded glm() function in R and the imported data. The resulted coefficients (the parameter θ for the logistic regression model) are taken for comparison with what we will manually calculated in the following codes. It takes four iterations for the glm() model to converge. Subsequently, we supplied the starting value theta0 = c(0, 0, 0, 0), along with the imported data, to the NR() function that we have built.

Code :

```
##
## #---- Load data for task 2 ----
## library(RCurl)
## set.seed(900101)
## länk <- "https://raw.githubusercontent.com/mskoldSU/MT5003_HT17/master/Projekt/proj_data.csv"
## data_individ <- read.csv(text = getURL(länk))
## idx <- sample(1:nrow(data_individ), 1000)
## data_individ <- data_individ[idx, ]
## save(data_individ, file = "proj_data.Rdata")
##
##
## #---- Task-2 ----
## modell <- glm(Resultat ~ Alder + Kon + Utbildare, data = data_individ, family = "binomial")
## summary(modell)
##
## y <- matrix(data_individ$Resultat, ncol=1)
## X <- model.matrix(Resultat ~ Alder + Kon + Utbildare, data = data_individ)
## theta0 = c(0, 0, 0, 0)
##
## theta4 <- NR(theta0, 4, y, X)
## compare_theta4 <- data.frame("theta_NR4" = theta4, "theta_R_model" = coef(modell))
##
## theta3 <- NR(theta0, 3, y, X) # We will use this value in the following steps.
## compare_theta3 <- data.frame("theta_NR3" = theta3, "theta_R_model" = coef(modell))
##
## theta2 <- NR(theta0, 2, y, X)
## compare_theta2 <- data.frame("theta_NR2" = theta2, "theta_R_model" = coef(modell))
```

Output :

We started from doing four iterations as what R needs, and the results are exactly same as the estimates from R (with eight digits of accuracy).

```
## Iteration 1 : theta = 0.08558482 -0.03220501 0.3624826 0.9754583
## Iteration 2 : theta = 0.1311712 -0.03592505 0.3995438 1.021972
## Iteration 3 : theta = 0.1320649 -0.035988 0.4000882 1.022574
## Iteration 4 : theta = 0.1320652 -0.03598801 0.4000883 1.022574
```

```
##                        theta_NR4 theta_R_model
## (Intercept)           0.13206518    0.13206518
## Alder                -0.03598801   -0.03598801
## KonMan                0.40008830    0.40008830
## UtbildareTrafikskola  1.02257373    1.02257373
```

So, we reduce the iteration to three to see the minimum sufficiency. The result is still very good.

```
## Iteration 1 : theta = 0.08558482 -0.03220501 0.3624826 0.9754583
## Iteration 2 : theta = 0.1311712 -0.03592505 0.3995438 1.021972
## Iteration 3 : theta = 0.1320649 -0.035988 0.4000882 1.022574
```

```
##                       theta_NR3 theta_R_model
## (Intercept)           0.1320649    0.13206518
## Alder                -0.0359880   -0.03598801
## KonMan                0.4000882    0.40008830
## UtbildareTrafikskola  1.0225736    1.02257373
```

We then further reduced the number of the iteration to two, the result looks like the following:

```
## Iteration 1 : theta = 0.08558482 -0.03220501 0.3624826 0.9754583
## Iteration 2 : theta = 0.1311712 -0.03592505 0.3995438 1.021972
```

```
##                         theta_NR2 theta_R_model
## (Intercept)            0.13117119    0.13206518
## Alder                 -0.03592505   -0.03598801
## KonMan                 0.39954383    0.40008830
## UtbildareTrafikskola  1.02197231    1.02257373
```

Observation :

Other than the coefficient for "KonMan", the rest have still two digits of accuracy. But, for all coefficients to have two digits of accuracy, we keep the number of iteration to three.

Task 3

Approach :

To calculate the standard errors from the ML estimates, the formula $\sqrt{\text{diag}(I(\hat{\theta})^{-1})}$ was applied where I($\hat{\theta}$) represents the Fisher Information Matrix. These values were then compared with the standard errors from R in the provided summary.

Code :

```
## #---- Task-3 ----
## # Compare the standard error values with R
##
## info <- I(theta3, y, X) #dim = k*k
## info_inv <- solve(info)   #dim = k*k
## diagonal <- diag(info_inv) # dim = k*1
## standard_errors <- sqrt(diagonal) #dim = k*1
##
##
## # Standard errors from R in summary
## r_se <- summary(modell)$coefficients[, "Std. Error"]
##
##
## # Creating a comparison data frame
## comparison_se <- data.frame(
##   "Se_R_model" = r_se,
##   "Se_computed" = standard_errors
## )
```

Output :

A comparison table between the standard errors in the provided R summary and the errors from ML estimates are attached for further reference.

```
comparison_se
```

```
##                         Se_R_model Se_computed
## (Intercept)            0.246637988 0.246638043
## Alder                  0.008813442 0.008813445
## KonMan                 0.136349528 0.136349561
## UtbildareTrafikskola 0.158272498 0.158272528
```

Observation :

The standard errors derived from the ML Estimates, computed using the Fisher information matrix, align with the standard errors provided in the R summary. The values are consistent up until the 6th decimal place. This suggests that the Fisher information matrix is reliable and R is using the Fisher information method for computing the standard errors.

Task 4

Approach :

Re-sample the response variable (y_boot) : In each iteration, the response variable y_boot is re-sampled based on the predicted probabilities p_hat. This re-sampling is done using a Bernoulli distribution, so the values of y_boot will vary across iterations.

Bootstrap model fitting (model_boot) : In each iteration, the y_boot changes, and the logistic regression model model_boot which is fitted using y_boot varies as well. The model's coefficients will differ slightly because the response data is changing.

Bootstrap Coefficients (bootstrap_estimates) : These are the theta estimates calculated for each re-sample. The coefficients of the logistic regression model (bootstrap_estimates), obtained by fitting model_boot, change in each iteration because the model is fitted on a re-sampled version of the response variable.

Predicted probabilities (p_bootstrap) : Since the model coefficients change in each iteration, the predicted probabilities, which come from *predict(model_boot, newdata = new_data, type = "response")*, will also vary for each iteration.

Bootstrap Standard Errors (bootstrap_se) : The standard errors are computed for each re-sample. A comparison table summarizing the result is attached for further reference.

95% Confidence Interval Construction (bootstrap_ci) : A 95% confidence interval is computed for the use case mentioned of the same age woman with a private education of their own.

The sample size of bootstrap (n_bootstrap) : The number of bootstrap samples to be generated. We have chosen 10,000 for the calculation of standard errors and for the calculation of the 95% confidence interval. According to the book, when the sample size is big enough (over 10,000), the results should be very close to the estimated values from the original sample. The standard error computed with 10,000 bootstrap sample size has a more accurate prediction of the standard errors displayed in the R summary and the computed values from ML estimates.

Code :

```
## #---- Task-4 ----
##
## # Bootstrap function for the SE
## bootstrappingSE <- function(f_hat, n){
##   # Matrix to store the bootstrap estimates for each coefficient
##   bootstrap_estimates <- matrix(NA, nrow = n, ncol = length(coef(f_hat)))
##
##   for (i in 1:n) {
##     # Step 1: Get the predicted probabilities
##     p_hat <- predict(f_hat, type = "response")
##
##     # Step 2: Resample y based on the predicted probabilities
##     y_boot <- rbinom(n = length(p_hat), size = 1, prob = p_hat)
##
##     # Step 3: Refit the logistic model using the bootstrapped y
##     model_boot <- glm(y_boot ~ Alder + Kon + Utbildare, data = data_individ, family = binomial)
##
##     # Step 4: Store the bootstrapped estimates
##     bootstrap_estimates[i, ] <- coef(model_boot)
##   }
##   return(bootstrap_estimates)
## }
##
## # Calculate Standard Errors
## n_bootstrap <- 10000
## bootstrap_estimates <- bootstrappingSE(modell, n_bootstrap)
##
## # Calculate the standard deviation of the bootstrapped coefficients (standard errors)
## bootstrap_se <- apply(bootstrap_estimates, 2, sd)
##
## # Add the bootstrap standard errors to the comparison matrix
## comparison_se$Se_bootstrap <- bootstrap_se
##
##
## # Bootstrap 95% confidence interval for the probability that someone privately educated,
## #of age 34, female is successful
## bootstrappingP <- function(f_hat, new_data, n){
##   # Vector to store the bootstrap probabilities
##   p_bootstrap <- numeric(n)
##
##   for (i in 1:n) {
##     # Step 1: Get the predicted probabilities
##     p_hat <- predict(f_hat, type = "response")
##
##     # Step 2: Resample y based on the predicted probabilities
##     y_boot <- rbinom(n = length(p_hat), size = 1, prob = p_hat)
##
##     # Step 3: Refit the logistic model using the bootstrapped y
##     model_boot <- glm(y_boot ~ Alder + Kon + Utbildare, data = data_individ, family = binomial)
##
##     # Step 4: Predict for the specific individual using the bootstrapped model
##     p_bootstrap[i] <- predict(model_boot, newdata = new_data, type = "response")
##   }
##   return(p_bootstrap)
## }
##
##
## X_new <- data.frame("Alder" = 34, "Kon"="Kvinna", "Utbildare"="Privatist")
## n_bootstrap_p <- 10000
## p_bootstrap <- bootstrappingP(modell, X_new, n_bootstrap_p)
##
## # Calculate the 95% Confidence Interval
## bootstrap_ci <- quantile(p_bootstrap, c(0.025, 0.975))
```

Output :

A comparison table between the standard errors in the provided R summary, the errors from ML estimates and the errors from bootstrap samples are attached for further reference.

```
comparison_se
```

```
##                         Se_R_model Se_computed Se_bootstrap
## (Intercept)          0.246637988 0.246638043  0.250256521
## Alder                0.008813442 0.008813445  0.008921794
## KonMan               0.136349528 0.136349561  0.138668747
## UtbildareTrafikskola 0.158272498 0.158272528  0.158454803
```

The limits of the 95% bootstrap confidence interval for the probability that someone privately educated of your own age and sex is successful is attached here.

```
bootstrap_ci
```

```
##      2.5%     97.5%
## 0.2003693 0.3027108
```

Observation :

- The standard errors derived from the bootstrap estimates, align with the standard errors provided in the R summary and the computed Standard Errors from ML estimates. The values are consistent up until the 3th decimal place. This suggests that the bootstrap estimates are reliable.