03_question3

44

2022-06-02

Q3 Assume that (Y_i, X_i), $i = 1, \ldots, n$ are independent observations, where Y_i is binary and X_i is a d–dimensional vector of covariates.

We wish to fit a logistic regression model of Y to X, i.e., $P(Y_i = 1) = \frac{exp(X_i\beta)}{(1+exp(X_i\beta))}$ where β are the regression coefficients.

Note that we are fitting a model without an intercept

(a). Determine a generative model based on the multivariate normal distribution

for $X|Y$ with $d = 3$ and $n = 102$ that leads to estimates of β that are either negative infinity or infinity. We will refer to this as a nonconvergence situation.

List out the following values: the mean vector and variance-covariance matrix of $X|Y$ and p, the probability of $Y = 1$.

$$
\begin{aligned}
Y_i|X_i &\sim Bernoulli(p_i) \\
logit(E[Y_i]) &= logit(p_i) = X_i\beta \\
p_i(X_i) &= \frac{\exp(X_i\beta)}{1+\exp(X_i\beta)}
\end{aligned}
$$

$$
\begin{aligned}
f(x, y|\beta) &= L(\beta; X, Y) \\
&= \prod_{i=1}^{n} (p_i(x_i))^{y_i}(1 - p_i(x_i))^{1-y_i} \\
&= \prod_{i=1}^{n} (\frac{\exp(X_i\beta)}{1+\exp(X_i\beta)})^{y_i}(1 - \frac{\exp(X_i\beta)}{1+\exp(X_i\beta)})^{1-y_i} \\
logL(\beta; X, Y) &= \sum_{i=1}^{n} log(1 - \frac{\exp(X_i\beta)}{1+\exp(X_i\beta)}) + \sum_{i=1}^{n} y_i X_i\beta \\
\frac{\nabla logL(\beta; X, Y)}{\nabla\beta} &= \sum_{i=1}^{n} (y_i - \frac{\exp(X_i\beta)}{1+\exp(X_i\beta)})X_i \overset{set}{=} 0 \\
\forall i, \quad y_i &= \frac{\exp(X_i\hat{\beta})}{1+\exp(X_i\hat{\beta})}
\end{aligned}
$$

In a general situation, we will not be able to solve exactly to get a closed form solution, because binary outcome y_i can not be estimate by $p_i(X_i) = \frac{\exp(X_i\beta)}{1+\exp(X_i\beta)}$. For most of the time, numerical methods must be applied. However, in some rare cases (β is noncovergence, as $-\infty$ or ∞), $p_i(X_i) = \frac{\exp(X_i\beta)}{1+\exp(X_i\beta)}$ can take the exact value of y_i as either 0 or 1. This means the value of Y can separate a given X completely. Then we will have a situation of complete separation. In such case, the more extreme the value of β takes, the larger the likelihood is. Hence the maximum likelihood estimates on the parameter β does not exist but take the value of $-\infty$ or ∞.

the mean vector and variance-covariance matrix of $X|Y$ and p, the probability of $Y = 1$.

I will set $x1$ is correlated with y with mean $E[x_1] = 0$, and make sure $\mathbf{X\beta} = 0$. In such case, we have $logit(p) = \mathbf{X\beta} = 0$. So we set $\mu_1 = E[X_1] = 0$, and $E[x_2 * \beta_2 + x_3 \times \beta_3] = 0$ to generate $Y_i \sim Bernoulli(0.5)$. Then we set x_1 perfectly complete separated by Y_i, such that when $P(X_1 > 0|Y = 1) = 1$.

$$p = E[Y] = 0.5$$
$$logit(p) = \mathbf{X\beta} = 0$$
$$\boldsymbol{\beta} = (1, 0.5, -0.25)$$
$$\mathbf{X} \sim MVN(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$
$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 = 0 \\ \mu_2 = 1 \\ \mu_3 = 2 \end{pmatrix}, \ \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 = 0.25 & 0 & 0 \\ 0 & \sigma_2^2 = 0.25 & 0 \\ 0 & 0 & \sigma_3^2 = 0.25 \end{pmatrix}$$

(b). Give a 2-3 sentence description which describes when we can expect nonconvergence.

The outcome variable separates certain predictor variables completely and almost perfectly. There is no pair (Y, X) with give Y value and X belongs to certain region, such that the frequency is zero in the sample, for example $P\big(Y = 1|X_{sub} \in (-\infty, 0)\big) = 0$ and $P\big(Y = 1|X_{sub} \in (0, \infty)\big) = 1$. In such situation, maximum likelihood estimates on the parameter β does not exist.

(c). Give a 2-3 sentence description which describes how nonconvergence relates to multicollinearity.

In both situations, the model does not have reasonable estimates for predictor variables but remains valid prediction for the outcome. The noncovergence situation is a special type of collinearity, after link function transformation on y, there is a linear relationship with X_2, X_3, $logit\big(P(Y)\big) = x_2\beta_2 + x_3\beta_3$. Meanwhile, we know $Y = I(X_1 \in S)$ for separation case. Hence $logit\big(P(X_1 \in S)\big) = x_2\beta_2 + x_3\beta_3$ is collinearized with X belongs to certain set.

(d). Modify the model in (a) to achieve convergence.

Based on the modified model, fit a logistic regression of y on X using 100000 observations from $y = 1$ and 100000 observations from $y = 0$.

Report the estimated regression coefficients and standard errors.

Assuming $E[Y] = \hat{p} = 0.5$, such that $logit(\hat{p}) = X\beta = 0$, we can discard the predictor x_1, which is correlated with outcomes Y. If we only use the other two predictors x_2, x_3, the model should converge.

	beta1			beta2			beta3		
	mean	bias	sd	mean	bias	sd	mean	bias	sd
Nonconvergence	-24134.70	24135.70	3196.21	0.07	0.43	2.09	-0.03	0.22	1.22
Convergence				0.00	0.50	0.01	0.00	0.25	0.00

As we can see in the table, the estimates indeed converge with "reasonable" point estimates and standard deviations, after we remove the predictor cause complete separation. For the nonconvergence situation, the point estimate takes extreme values and large standard deviation.

For the convergence model, even though we get a reasonable estimate with standard deviations, the estimates are very close to zero, not exactly the values used for estimation.

```
library(tidyverse)
set.seed(55555)

n <- 100000
N <- n * 10
beta1 <- 1
beta2 <- 0.5
beta3 <- -0.25

x1abs <- c(abs(rnorm(n, mean = 0, sd = 0.5)))
x2 <- c(rnorm(2 * N, mean = 1, sd = 0.5))
x3 <- c(rnorm(2 * N, mean = 2, sd = 0.5))

beta23 <- as.matrix(c(beta2, beta3))
X23 <- as.matrix(cbind(x2, x3))
# str(beta23); dim(beta23)
# str(X23); dim(X23)

logit_p <- X23 %*% beta23
mean(logit_p) ## [1] 4.258361e-05
p_hat <- 1 / (1 + exp(-logit_p))
mean(p_hat) ## [1] 0.500011

y <- map(p_hat,
         ~rbinom(n = 1, size = 1, prob = .)) %>%
  unlist()

data_3_0 <- data.frame(y, x2, x3) %>%
  ## simulate x1 which can be separate by outcome y completely
  mutate(x1 = ifelse(y == 1, x1abs, -x1abs))
# View(data_3_0)

fit_3_0 <- glm(y ~ 0 + x1 + x2 + x3,
```

```r
              family = binomial("logit"),
              data = data_3_0)

n <- 100000
beta1 <- 1
beta2 <- 0.5
beta3 <- -0.25
y <- c(rep(0, n), rep(1, n))
x1 <- c(abs(rnorm(n, mean = , sd = 0.5)),
        -abs(rnorm(n, mean = 1, sd = 0.5)))
x2 <- c(rnorm(2 * n, mean = 1, sd = 0.5))
x3 <- c(rnorm(2 * n, mean = 2, sd = 0.5))

data_3_1 <- data.frame(y, x1, x2, x3)
# mean(data_3_1$x1)
# mean(data_3_1$y)
# save(data_3_0, data_3_1, file = "bios_qexam_data_3_0.Rdata")

n <- 100000
beta1 <- 1
beta2 <- 0.5
beta3 <- -0.25
y <- c(rep(0, n), rep(1, n))
glm_con <- matrix(NA, nrow = n, ncol = 3)
glm_unc <- matrix(NA, nrow = n, ncol = 2)

for (i in seq_along(1:1000)) {
  x1 <- c(abs(rnorm(n, mean = 1, sd = 0.5)),
          -abs(rnorm(n, mean = 1, sd = 0.5)))
  x2 <- c(rnorm(2 * n, mean = 1, sd = 0.5))
  x3 <- c(rnorm(2 * n, mean = 2, sd = 0.5))

  data_sim <- data.frame(y, x1, x2, x3)

  fit_3_0 <- glm(y ~ 0 + x1 + x2 + x3,
                 family = binomial("logit"),
                 data = data_sim)

  fit_3_1 <- glm(y ~ 0 + x2 + x3,
                 family = binomial("logit"),
                 data = data_sim)

  glm_con[i, ] <- fit_3_0$coefficients
  glm_unc[i, ] <- fit_3_1$coefficients
}

save(glm_con, glm_unc,
     file = "bios_qexam_data_3_0.Rdata")

load("bios_qexam_data_3_0.Rdata")

library(matrixStats)
colMeans(as.matrix(glm_con), na.rm = TRUE)

colMeans(as.matrix(glm_unc), na.rm = TRUE)

colMeans(as.matrix(glm_con), na.rm = TRUE) - c(1, 0.5, -0.25)

colMeans(as.matrix(glm_unc), na.rm = TRUE) - c(1.5, -0.25)

colSds(as.matrix(glm_con), na.rm = TRUE)

colSds(as.matrix(glm_unc), na.rm = TRUE)
```

(e). Perform a simulation study using the generative model in (a). comparing different penalization schemes. As in (a), we will assume a sample size of 103.

Please use the following methods: (1) LASSO regression (using the glmnet package in R); (2) Firth correction (using the logistf package in R).

For both methods, you will actually fit the model with the intercept. Thus, we are fitting a misspecified model, and the goal is to see how the corrections fare in terms of bias.

For the simulation study, specify the following: - Tuning parameters used for the two penalized regressions; - The number of simulated datasets being used. - Report on the mean bias for the parameters using each of the methods.

(NB: For the LASSO, you will get a warning message of the form "In lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, … : one multinomial or binomial class has fewer than 8 observations; dangerous ground"; you can ignore that. For the Firth correction, you will get warning messages of the form "fitted probabilities numerically 0 or 1 occurred for variable V2"; you can disregard those. In the event that any of your simulation values do not converge, you may remove those from the simulation study. Finally, for the Firth correction, you will want to set the maximum number of iterations to be at least 10000).

The tuning parameter used for glmnet $\alpha = 1$ as LASSO, and $\lambda = 0.1,0.2, and 0.5$ The tuning parameter used for logsitf $\alpha = 1$ as LASSO, and $\lambda = 0.1,0.2, and 0.5$. The sample size is 103 for each dataset, which performed 1000 iterations.

$$L^*\beta = L(\beta) \times det\big(I(\beta)\big)^{1/2}$$

where $I(\beta)$ is the Fisher information matrix and $L(\beta)$ is the likelihood. Firth-type penalization removes the first-order bias of the MLE and bias-preventive rather than corrective for rare events. I do not exactly understand why we need to use Firth's method on noncovergence situation. There is no tuning parameter.

As shown in the table below, both methods can provide reasonable and converged estimations. Because the model fitted with LASSO and Firth penalization are mis-specified with intercept. I am not sure whether the mean bias is meaningful comparing with the true data generating parameters (not shown in table below). Instead, we calculated the mean value over three lambda setting, to see how the corrections fare in terms of bias from the expected mean. As we can see, with a larger penalized parameter lambda, LASSO quickly eliminate the number of parameters and the estimation is also shrinking down to zero quickly. However, for the Firth's penalization, the larger lambda value affects each beta in similar pattern: although the estimation of every beta shrinks down, none of the estimation shrinks down to zero. Overall, LASSO change the estimation dramatically over lambda value than Firth method.

Type		glmnet			logisft		
lambda		0.100	0.200	0.500	0.100	0.200	0.500
beta0 = 0	Mean	0.009	0.007	0.005	0.009	0.014	0.010
	SD	0.184	0.175	0.206	2.473	2.461	2.286
	bias	0.002	0.000	-0.002	-0.002	0.003	-0.001
beta1 = 1	Mean	3.899	1.912		32.734	32.725	30.975
	SD	0.305	0.171		5.129	5.106	4.703
	bias	1.962	-0.025		0.589	0.580	-1.170
beta2 = 0.5	Mean				0.014	0.015	0.017
	SD				1.082	1.083	1.045
	bias				-0.001	0.000	0.002
beta3 = -0.25	Mean				0.001	-0.004	-0.004
	SD				1.076	1.066	0.977
	bias				-0.001	-0.006	-0.006

```
library(tidyverse)

sim_103 <- function(n = 103, seed){
  set.seed(seed)
  x1 <- c(abs(rnorm(n, mean = 0, sd = 0.5)))
  x2 <- c(rnorm(n, mean = 1, sd = 0.5))
  x3 <- c(rnorm(n, mean = 2, sd = 0.5))
  logit_p <- 0
  p_hat <- 1 / (1 + exp(-logit_p))
  y <- map(rep(p_hat, n),
         ~rbinom(n = 1, size = 1, prob = .)) %>%
  unlist()
  data_3 <- data.frame(y, x2, x3) %>%
  ## simulate x1 which can be separate by outcome y completely
  mutate(x1 = ifelse(y == 0, -x1, x1))

  return(data_3)
}

sim_103s <- map(1:1000, ~sim_103(seed = .))

library(glmnet)

glmnet_01 <- map_dfc(sim_103s,
               ~glmnet(x = .[, c(4, 2, 3)],
                       y = .[, 1],
                       family = "binomial",
                       alpha = 1,
                       lambda = 0.1) %>%
                 coef(s = 0.1) %>%
                 as.vector())

glmnet_02 <- map_dfc(sim_103s,
               ~glmnet(x = .[, c(4, 2, 3)],
                       y = .[, 1],
                       family = "binomial",
                       alpha = 1,
                       lambda = 0.2) %>%
                 coef(s = 0.2) %>%
                 as.vector())


glmnet_05 <- map_dfc(sim_103s,
               ~glmnet(x = .[, c(4, 2, 3)],
                       y = .[, 1],
                       family = "binomial",
                       alpha = 1,
                       lambda = 0.5) %>%
```

```
                coef(s = 0.5) %>%
                as.vector())

save(glmnet_01, glmnet_02, glmnet_05,
     file = "bios_qexam_question_3_glmnet.Rdata")

load("bios_qexam_question_3_glmnet.Rdata")
library(matrixStats)
rowMeans(glmnet_01)

## [1] 0.008867274 3.899150826 0.000000000 0.000000000

rowMeans(glmnet_02)

## [1] 0.007041042 1.911596483 0.000000000 0.000000000

rowMeans(glmnet_05)

## [1] 0.00505556 0.00000000 0.00000000 0.00000000

rowSds(as.matrix(glmnet_01))

## [1] 0.1835924 0.3046741 0.0000000 0.0000000

rowSds(as.matrix(glmnet_02))

## [1] 0.1745398 0.1709125 0.0000000 0.0000000

rowSds(as.matrix(glmnet_05))

## [1] 0.2062338 0.0000000 0.0000000 0.0000000

library(logistf)
logistf_01 <- map_dfc(sim_103s,
               ~logistf(data = .,
                    formula = y ~ 1 + x1 + x2 + x3,
                    control = logistf.control(maxit = 1000),
                    modcontrol = logistf.mod.control(tau = 0.1),
                    firth = TRUE) %>%
                 coef() %>%
                 as.vector())

logistf_02 <- map_dfc(sim_103s,
               ~logistf(data = .,
                    formula = y ~ 1 + x1 + x2 + x3,
                    control = logistf.control(maxit = 1000),
                    modcontrol = logistf.mod.control(tau = 0.2),
                    firth = TRUE) %>%
                 coef() %>%
                 as.vector())

logistf_05 <- map_dfc(sim_103s,
               ~logistf(data = .,
                    formula = y ~ 1 + x1 + x2 + x3,
                    control = logistf.control(maxit = 1000),
                    modcontrol = logistf.mod.control(tau = 0.5),
                    firth = TRUE) %>%
                 coef() %>%
                 as.vector())

save(logistf_01, logistf_02, logistf_05, file = "bios_qexam_question_3_logistf.Rdata")

load("bios_qexam_question_3_logistf.Rdata")
library(matrixStats)
rowMeans(logistf_01)

## [1] 8.943984e-03 3.273415e+01 1.358531e-02 6.528317e-04
```

```
rowMeans(logistf_02)

## [1]  0.014238817 32.725466962  0.014733654 -0.003744427

rowMeans(logistf_05)

## [1]  0.009988114 30.974600643  0.017037162 -0.003669026

rowSds(as.matrix(logistf_01))

## [1] 2.473284 5.128945 1.082139 1.075547

rowSds(as.matrix(logistf_02))

## [1] 2.461443 5.105644 1.083050 1.066187

rowSds(as.matrix(logistf_05))

## [1] 2.2856060 4.7032294 1.0445197 0.9766652
```