Assignment4

Student number:1407870

2023-10-22

Subject name: A First Course In Statistical Learning
Subject code: MAST90104

Q1

(a)

```
wine=read.csv('/Users/guanmuhan/Downloads/winequality-red.csv',sep = ';')
wine$quality = factor(wine$quality)
levels(wine$quality) <- c("bad","bad","average","average" , "good","good")
library(nnet)
mmod=multinom(quality~.,wine)
```

```
## # weights:  39 (24 variable)
## initial  value 1756.681050
## iter  10 value 1049.635007
## iter  20 value 689.921334
## iter  30 value 653.028931
## iter  40 value 651.960731
## iter  50 value 651.732058
## iter  60 value 651.695020
## iter  70 value 651.546418
## iter  80 value 651.305172
## iter  90 value 648.593999
## final  value 647.675831
## converged
```

```
summary(mmod)
```

```
## Call:
## multinom(formula = quality ~ ., data = wine)
##
## Coefficients:
##         (Intercept) fixed.acidity volatile.acidity citric.acid residual.sugar
## average   -178.8060   -0.26900223        -4.462724  -1.2742330   -0.259419065
## good       122.3626    0.05907234        -6.782318  -0.6097511    0.009571902
##          chlorides free.sulfur.dioxide total.sulfur.dioxide   density        pH
## average  -5.485951          0.01736726         0.0161405294  197.9551 -4.534165
## good    -13.977148          0.02721645        -0.0004499205 -119.7577 -3.941803
##         sulphates   alcohol
## average  1.199091 0.4310564
## good     5.012159 1.1267605
##
## Std. Errors:
##         (Intercept) fixed.acidity volatile.acidity citric.acid residual.sugar
## average    2.705097     0.1543634         0.835823    1.218232     0.08282364
## good       3.129535     0.1706778         1.110980    1.444455     0.09777426
##         chlorides free.sulfur.dioxide total.sulfur.dioxide  density       pH
## average  3.101619          0.02292630          0.008267819 2.646911 1.391239
## good     4.459550          0.02565747          0.009496124 3.052356 1.600228
##         sulphates   alcohol
## average  1.269845 0.1758899
## good     1.358037 0.1949868
##
## Residual Deviance: 1295.352
## AIC: 1343.352
```

```r
mmodi=step(mmod,trace = F)
```

```
## trying - fixed.acidity
## trying - volatile.acidity
## trying - citric.acid
## trying - residual.sugar
## trying - chlorides
## trying - free.sulfur.dioxide
## trying - total.sulfur.dioxide
## trying - density
## trying - pH
## trying - sulphates
## trying - alcohol
## # weights:  36 (22 variable)
## initial  value 1756.681050
## iter  10 value 962.548544
## iter  20 value 660.650129
## iter  30 value 652.962534
## iter  40 value 652.709099
## iter  50 value 652.619043
## iter  60 value 652.562298
## iter  70 value 652.480458
## iter  80 value 651.061574
## iter  90 value 648.331708
## final  value 648.331656
## converged
## trying - fixed.acidity
## trying - volatile.acidity
## trying - citric.acid
## trying - residual.sugar
## trying - chlorides
## trying - total.sulfur.dioxide
## trying - density
## trying - pH
## trying - sulphates
## trying - alcohol
```

```r
summary(mmodi)
```

```
## Call:
## multinom(formula = quality ~ fixed.acidity + volatile.acidity +
##     citric.acid + residual.sugar + chlorides + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol, data = wine)
##
## Coefficients:
##         (Intercept) fixed.acidity volatile.acidity citric.acid residual.sugar
## average   -177.1713   -0.26252634        -4.565347  -1.4550010    -0.24141265
## good       125.7384    0.07207673        -6.952064  -0.8516209     0.02213188
##          chlorides total.sulfur.dioxide   density        pH sulphates   alcohol
## average  -5.434441          0.020732275  196.0641 -4.486351  1.255191 0.4420856
## good    -13.805282          0.006790962 -123.6146 -3.799881  5.050087 1.1345209
##
## Std. Errors:
##         (Intercept) fixed.acidity volatile.acidity citric.acid residual.sugar
```

```
## average    2.704340     0.1545768        0.8275218    1.201010     0.07916868
## good       3.125144     0.1708142        1.1010330    1.427641     0.09566638
##         chlorides total.sulfur.dioxide  density       pH sulphates   alcohol
## average  3.111854          0.006059025 2.649079 1.393743  1.273329 0.1756397
## good     4.468744          0.006894894 3.048687 1.597277  1.360258 0.1946937
##
## Residual Deviance: 1296.663
## AIC: 1340.663
```

```
#We can see there are eleven beat_is for each tow of the outcomes
```

(b)

```
#First we convert quality into an ordered factor
wine$quality=ordered(wine$quality, levels=c("bad", "average", "good"))
library(MASS)
omod=polr(quality~.,data = wine)
summary(omod)
```

```
##
## Re-fitting to get Hessian
```

```
## Call:
## polr(formula = quality ~ ., data = wine)
##
## Coefficients:
##                         Value Std. Error   t value
## fixed.acidity          0.098341   0.068831   1.42874
## volatile.acidity      -3.879626   0.544055  -7.13094
## citric.acid            0.053926   0.660585   0.08163
## residual.sugar         0.091284   0.054391   1.67830
## chlorides             -6.706752   1.966230  -3.41097
## free.sulfur.dioxide    0.004318   0.009655   0.44719
## total.sulfur.dioxide  -0.003905   0.003244  -1.20383
## density              -86.533740   1.361587 -63.55357
## pH                    -1.523710   0.709843  -2.14654
## sulphates              2.683266   0.457355   5.86692
## alcohol                0.774485   0.078961   9.80847
##
## Intercepts:
##              Value    Std. Error t value
## bad|average  -87.1941   1.3953   -62.4934
## average|good -80.5404   1.4091   -57.1567
##
## Residual Deviance: 1373.733
## AIC: 1399.733
```

```
omodi=step(omod,trace=F)
summary(omodi)
```

```
##
## Re-fitting to get Hessian
```

```
## Call:
## polr(formula = quality ~ volatile.acidity + chlorides + pH +
##     sulphates + alcohol, data = wine)
```

```
##
## Coefficients:
##                    Value Std. Error t value
## volatile.acidity -4.0170     0.4688  -8.569
## chlorides        -6.5630     1.8940  -3.465
## pH               -2.1087     0.5221  -4.039
## sulphates         2.4779     0.4547   5.450
## alcohol           0.8712     0.0758  11.494
##
## Intercepts:
##              Value   Std. Error t value
## bad|average  -3.0044  1.7780    -1.6898
## average|good  3.6146  1.7748     2.0366
##
## Residual Deviance: 1377.604
## AIC: 1391.604
```

There are much more parameters for the multinomial model than that for the ordinal model. Residual deviance higher for the ordinal model, and the df for the final ordinal model is 7 while the df for the final multinomial is 22 .But the AIC for multinomial model is slightly smaller than that for the ordinal regression model, so we prefer to choose it.

(c)

```
#newdata for two models
newdata_multinomial=matrix(c(1,7.9,0.4,0.2,1.7,0.1,36,0.997,3.3,0.9,10),11,1)
newdata_ordinal=matrix(c(0.4,0.1,3.3,0.9,10),5,1)
#coefficient matrixes for two moedls
(coefficients_multi=matrix(coef(mmodi),2,11) )
```

```
##           [,1]        [,2]      [,3]       [,4]        [,5]       [,6]
## [1,] -177.1713 -0.26252634 -4.565347 -1.4550010 -0.24141265  -5.434441
## [2,]  125.7384  0.07207673 -6.952064 -0.8516209  0.02213188 -13.805282
##             [,7]      [,8]      [,9]    [,10]     [,11]
## [1,] 0.020732275  196.0641 -4.486351 1.255191 0.4420856
## [2,] 0.006790962 -123.6146 -3.799881 5.050087 1.1345209
```

```
(coefficients_ordi=matrix(coef(omodi), 1,5) )
```

```
##           [,1]      [,2]      [,3]     [,4]      [,5]
## [1,] -4.016974 -6.563039 -2.108722 2.477854 0.8712256
```

```
#linear combinations for two models
linear_predictors_multinomial = coefficients_multi%*% newdata_multinomial
linear_predictors_ordinal=coefficients_ordi%*% newdata_ordinal
#linear combinations for multinomial regression
linear_predictors_multinomial
```

```
##          [,1]
## [1,] 4.651586
## [2,] 2.365144
```

```
#linear combinations for ordinal regression
linear_predictors_ordinal
```

```
##          [,1]
## [1,] 1.720449
```

```r
#inverse function for multinomial
softmax <- function(z) {
  exp_z <- exp(z)
  return(exp_z / sum(exp_z))
}
#probabilities for multinomial
probabilities_multinomial=softmax(c(0, linear_predictors_multinomial))
cat('By multimonial regression model, the probability of wine is bad:',probabilities_multinomial[1],'av
```

```
## By multimonial regression model, the probability of wine is bad: 0.008591316 average: 0.8999492 good
```

```r
#By ordinal regression
(g_r01=omodi$zeta[1]-linear_predictors_ordinal)
```

```
##           [,1]
## [1,] -4.724811
```

```r
(g_r02=omodi$zeta[2]-linear_predictors_ordinal)
```

```
##          [,1]
## [1,] 1.894104
```

```r
probabilities_ordinal_bad=1/(1+exp(-(g_r01)))
probabilities_ordinal_average=1/(1+exp(-(g_r02)))-probabilities_ordinal_bad
probabilities_ordinal_good=1-1/(1+exp(-(g_r02)))
cat('By ordinal regression model, the probability of wine is bad:',probabilities_ordinal_bad,'average:'
```

```
## By ordinal regression model, the probability of wine is bad: 0.00879436 average: 0.8604284 good: 0.13
```

(d)

$$odds\ ratio = \frac{\left[\frac{r_{12}}{1-r_{12}}\right]}{\left[\frac{r_{22}}{1-r_{22}}\right]} = \frac{e^{g(r_{12})}}{e^{g(r_{22})}} = e^{g(r_{12})-g(r_{22})} = e^{-(x_1-x_2)^T\beta} = e^{-(0.08-0.2)\beta_{chlorides}}$$

```r
difference=-(0.08-0.2)
odds_ratio=exp(difference*omodi$coefficients[2])
cat('the odds ratio of ordinal regression moedl',odds_ratio)
```

```
## the odds ratio of ordinal regression moedl 0.4549514
```

Q2

(a)

The likelihood is:

$$f(D|\beta) = \prod_i^n P(\epsilon_i|\beta) = (2\pi)^{-\frac{n}{2}} exp(-\frac{1}{2}\sum_{i=1}^n (y_i - \beta x_i)^2)$$

(b)

We have Bayes theorem, the posterior is still normal

$$posterior \propto likelihood \times prior$$

$$P(\beta|D) \propto P(D|\beta)P(\beta) = exp(-\frac{1}{2}\sum_i^n (y_i - \beta x_i)^2) \times exp(-\frac{1}{200}\beta^2) = exp(-\frac{1}{2}(\beta^2(\sum_i^n x_i^2 + \frac{1}{100}) - 2\beta\sum_i^n y_i x_i + \sum_i^n y_i^2))$$

A magnitude of $\frac{1}{100}$ is markedly small and can often be disregarded in equations. To elaborate, this suggests that when the variance of the prior probability is substantial, the information it contributes to the estimation of posterior distribution is minimal. Under these circumstances, the estimation of the posterior distribution is main dominated by the likelihood.

Therefore,we can get mean and variance of posterior distribution of β by $N(A, B)$,where $B = (\sum_i^n x_i^2)^{-1} = 0.04620728$, $A = B(\sum_i^n y_i x_i) = 1.94572$

```
data=read.csv("/Users/guanmuhan/Downloads/simplereg (2).csv")
(B=1/sum(data$x^2))
```

```
## [1] 0.04620728
```

```
(A=B*sum(data$x*data$y))
```

```
## [1] 1.94572
```

(c)

```
model <- lm(y ~  0+x , data=data) #no intercept
summary(model)#current beta is 1.94572
```

```
##
## Call:
## lm(formula = y ~ 0 + x, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6293 -0.4784 -0.1137  0.1198  1.5086
##
## Coefficients:
##   Estimate Std. Error t value Pr(>|t|)
## x   1.9457     0.1396   13.94  2.2e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6494 on 29 degrees of freedom
```

```
## Multiple R-squared:  0.8701, Adjusted R-squared:  0.8656
## F-statistic: 194.3 on 1 and 29 DF,  p-value: 2.195e-14
```

MH Algorithm

1. Initialize beta_current = 1.94572

2. FOR each iteration:

(a) Sample beta_new ~ N(beta_current, 0.25).

(b) Compute AR = P(beta_new | data) / P(beta_current | data).

(For normal distribution, the Q(beta|beta')/)Q(beta'|beta) is cancelled)

(c) Draw u ~ U(0,1).

(d) IF u < min(1, AR): beta_current = beta_new. Else: beta_current=beta_current

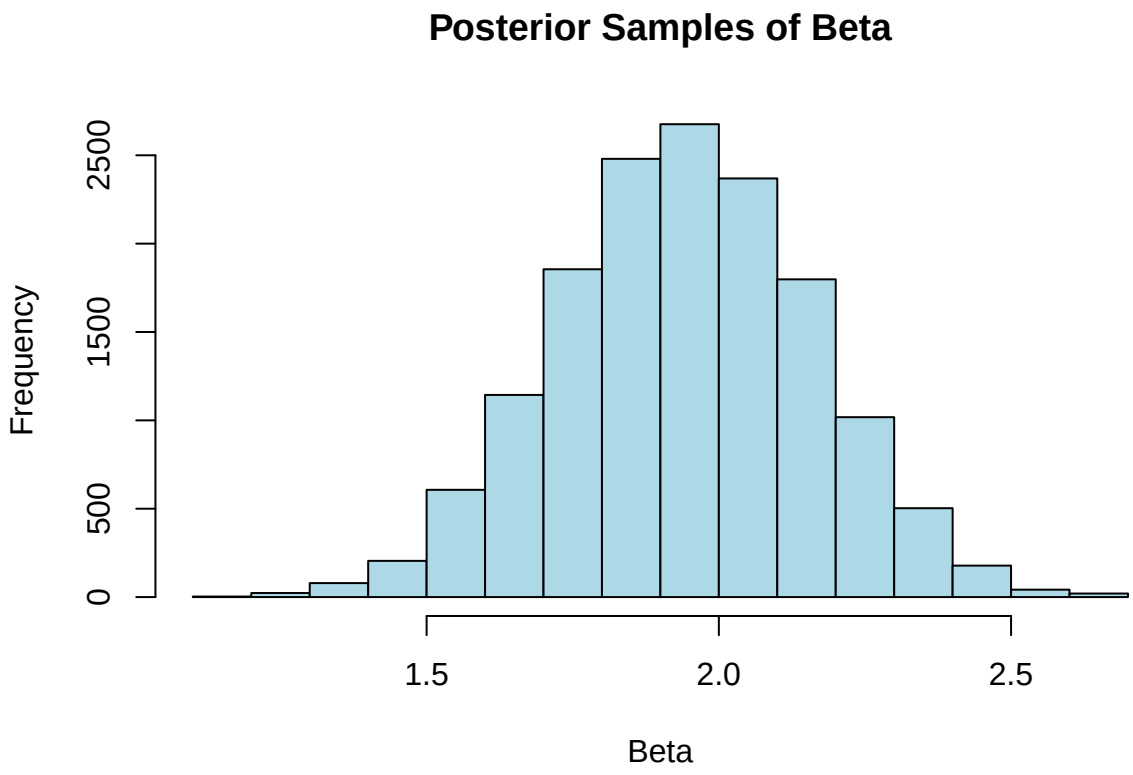
(e) Record beta_current.

3. take samples from the (T+1)-th iteration (throw out data generated in the burn-in stage)

(d)

```r
# Provided parameters and data
beta_current <- 1.94572
sigma_prior <- sqrt(100)  # std of priori dist
sigma_proposal <- 0.5     # std of proposal dist
iterations <- 20000       # number of iterations
burn_in <- 5000           # burn-in

#set seeds
set.seed(123)
# Metropolis-Hastings
beta_samples <- numeric(iterations - burn_in)
for (i in 1:iterations) {

  beta_new <- rnorm(1, mean = beta_current, sd = sigma_proposal)


  likelihood_current <- sum(dnorm(data$y - beta_current * data$x, mean = 0, sd = 1, log = TRUE))
  prior_current <- dnorm(beta_current, mean = 0, sd = sigma_prior, log = TRUE)

  likelihood_new <- sum(dnorm(data$y - beta_new * data$x, mean = 0, sd = 1, log = TRUE))
  prior_new <- dnorm(beta_new, mean = 0, sd = sigma_prior, log = TRUE)

  acceptance_ratio <- exp((likelihood_new + prior_new) - (likelihood_current + prior_current))

  # Accept or reject new beta values
  if (runif(1) < acceptance_ratio) {
    beta_current <- beta_new
  }

  # Store sampling results (remove burn-in period)
  if (i > burn_in) {
    beta_samples[i - burn_in] <- beta_current
  }
}
```

```r
hist(beta_samples, main = "Posterior Samples of Beta", xlab = "Beta", col = "lightblue", border = "blac
```




```r
(sample_mean <- mean(beta_samples))
```

```
## [1] 1.940294
```

```r
(sample_variance <- var(beta_samples))
```

```
## [1] 0.04789546
```

```r
#By comparison ,the sampling distribution is close to the result in part b
```