STAT534HW3

Bryan Ng, 2427348

2025-04-21

Problem 2

```
library(rcdd)
```

```
## If you want correct answers, use rational arithmetic.
## See the Warnings sections in help pages for
##     functions that do computational geometry.
```

```
isValidLogistic <- function(response,explanatory,data)
{
  if(0==length(explanatory))
  {
    return(TRUE);
  }
  logisticreg = suppressWarnings(glm(data[,response] ~ as.matrix(data[,as.numeric(explanatory)]),family=
  tanv = logisticreg$x;
  tanv[data[,response] == 1, ] <- (-tanv[data[,response] == 1, ]);
  vrep = cbind(0, 0, tanv);
  lout = linearity(vrep, rep = "V");
  return(length(lout)==nrow(data));
}

getLogisticAIC <- function(response, explanatory, data) {
  if (0 == length(explanatory)) {
    deviance = glm(data[, response] ~ 1,
                   family = binomial(link = "logit"))$deviance
  }
  else {
    deviance = glm(data[, response] ~ as.matrix(data[, as.numeric(explanatory)]),
                   family = binomial(link = "logit"))$deviance
  }
  return(deviance + 2 * (1 + length(explanatory)))
}

forwardSearchAIC <- function(V, response, data, last_B) {
  cand <- setdiff(V, last_B)
  curAIC <- getLogisticAIC(response, last_B, data)

  if (length(cand) == 0) return(last_B)
```

```
  bestAIC      <- curAIC
  bestAdd      <- NULL
  for (i in cand) {
    newA     <- c(last_B, i)
    newAIC   <- getLogisticAIC(response, newA, data)
    if (newAIC < bestAIC) {
      bestAIC  <- newAIC
      bestAdd  <- i
    }
  }
  if (!is.null(bestAdd)) {
    return(c(last_B, bestAdd))
  }
  else {
    return(last_B)
  }
}

get_valid_nbd_set <- function(V, explanatory, data, response) {
  add_one <- lapply(setdiff(V, explanatory), function(i) sort(c(explanatory, i)))
  delete_one <- lapply(explanatory, function(i) setdiff(explanatory, i))
  nbd <- c(add_one, delete_one)
  mask <- sapply(nbd, function(A) {
    isValidLogistic(response, A, data)
  })
  return(nbd[mask])
}

current_model_update <- function(A_prime, A_cur, B_cur, data, response) {
  V <- setdiff(seq_len(ncol(data)), response)
  B_greedy <- forwardSearchAIC(V, response, data, B_cur)

  AIC_A_prime <- getLogisticAIC(response, A_prime, data)
  AIC_B <- getLogisticAIC(response, B_greedy, data)

  if (AIC_A_prime < AIC_B) {
    B_next <- A_prime
  }
  else {
    B_next <- B_greedy
  }
  return(list(A = A_prime, B = B_next))
}

MC3_iter <- function(V, A_cur, B_cur, data, response) {
  nbd_valid <- get_valid_nbd_set(V, A_cur, data, response)
  index <- sample.int(length(nbd_valid), 1)
  A_prime <- nbd_valid[[index]]
  nbd_prime <- get_valid_nbd_set(V, A_prime, data, response)
  p_A_prime <- -getLogisticAIC(response, A_prime, data) - log(length(nbd_prime))
  p_A <- -getLogisticAIC(response, A_cur, data) - log(length(nbd_valid))

  if (p_A_prime > p_A || log(runif(1)) < (p_A_prime - p_A)) {
```

```
    res <- current_model_update(A_prime, A_cur, B_cur, data, response)
  }
  else {
    res <- list(A = A_cur, B = B_cur)
  }
  return(res)
}

MC3_search <- function(data, response, n_iter) {
  V <- setdiff(seq_len(ncol(data)), response)
  repeat {
    k    <- sample(length(V), 1)
    A0   <- sample(V, k)
    if (isValidLogistic(response, A0, data)) break
  }
  A_cur <- A0
  B_cur <- A0
  for (r in n_iter) {
    res    <- MC3_iter(V, A_cur, B_cur, data, response)
    A_cur  <- res$A
    B_cur  <- res$B
  }
  return(list(bestAICvars = sort(B_cur),
              bestAIC = getLogisticAIC(response, B_cur, data)))
}

path = "C:/Users/ncwbr/Desktop/534binarydata.txt"
data = as.matrix(read.table(path, header = FALSE))
set.seed(2427348)
response <- 61
n_iter <- 25
n_run <- 10
for (i in seq_len(n_run)) {
  cat("Chain", i, "\n")
  print(MC3_search(data, response, n_iter))
}
```

```
## Chain 1

## $bestAICvars
##  [1]  2  3 10 12 13 18 19 23 25 28 29 31 33 35 41 43 44 45 46 48 50 51 52 53 55
## [26] 57 58 59
##
## $bestAIC
## [1] 58
##
## Chain 2

## $bestAICvars
##  [1]  1  2  6  7  9 11 14 17 22 23 28 29 33 37 38 41 42 45 47 48 50 52 53 54 58
## [26] 59 60
##
```

```
## $bestAIC
## [1] 94.25605
##
## Chain 3
## $bestAICvars
## [1]  2  7 42 52 55
##
## $bestAIC
## [1] 145.1977
##
## Chain 4

## $bestAICvars
##  [1]  1  7  8 10 11 13 14 17 18 20 23 25 28 30 34 36 37 42 43 51 53 54 56 58
##
## $bestAIC
## [1] 50
##
## Chain 5
## $bestAICvars
##  [1]  2  3 11 21 24 27 28 33 45 49 51
##
## $bestAIC
## [1] 130.1401
##
## Chain 6
## $bestAICvars
##  [1]  1  7 13 17 23 24 28 30 32 45 47 48 51 52 56 58
##
## $bestAIC
## [1] 106.785
##
## Chain 7
## $bestAICvars
## [1]  4  6 13 23 33 45 48 52 60
##
## $bestAIC
## [1] 117.8966
##
## Chain 8
## $bestAICvars
##  [1]  7 10 12 13 14 16 37 38 42 43 45 47 48 50 51 55 58
##
## $bestAIC
## [1] 137.0361
##
## Chain 9
## $bestAICvars
##  [1]  2  4  8 11 17 19 20 21 22 25 31 38 40 41 44 55 59
##
## $bestAIC
## [1] 80.07676
##
## Chain 10
```

```
## $bestAICvars
##  [1]  5  7 12 14 16 18 23 25 28 30 33 41 45 52 53 54 55
##
## $bestAIC
## [1] 107.0496
```

With only 25 iterations of MC^3 algorithm, we found that the results exhibited substantial variability in both their best AIC scores and selected best models. Increasing the number of iterations might help the algorithm converge more reliably and yield a more stable result.