STAT534:Statistical Computing
Homework 3

Po-An, Chen(Andy Chen)

April 17, 2022

In this homework you will continue to work with logistic regressions and the dataset "534binarydata.txt". The forward and backward greedy algorithms from Homework *2* fail to properly solve the optimization problem

$$\min\{AIC(\mathcal{M}_A) : A \subseteq V\} \tag{1}$$

whose solution is the "best" logistic regression with respect to

$$AIC(\mathcal{M}_A) = D(\mathcal{M}) + 2 \cdot |\mathcal{M}|,$$

where $\mathcal{M}_A$ is the logistic regression with explanatory variables $A \subset V = \{1, 2, ..., p\}$. Remember that, in the dataset "534binarydata.txt", the first $p = 60$ columns correspond with the available p explanatory variables, while the last column corresponds with the binary response variable y.

The key drawback of the forward and backward greedy algorithms is related to their deterministic nature: at each step, the two procedures attempt to decrease the AIC of the current best logistic regression ny adding or deleting a variable. The algorithms stop when the addition or deletion of a variable fails to decrease the AIC. You will implement the following stochastic algorithm for solving the optimization problem (1). At each iteration, this new algorithm can decrease the AIC of the current regression, but it could also increase it. That is, the procedure could choose to move in a worse model to find its way to a better model. In the literature this algorithm is known as the Markov chain Monte Carlo model composition (MC^3) algorithm and it is a good wxample of a Metropolis-Hastings algorithm.

The MC^3 algorithm has been applied in many contexts to solve the problem of model selection\determination. The version of the MC^3 you will develop will have a novel feature: the algorithm must avoid visiting logistic regressions in which the MLEs do not exist, which implies that the corresponding numerical values of the AIC or BIC are flawed. These flawed numerical values could be smaller than the solution of the optimization problem (1). Thus,

if invalid logistic regressions are not avoided, the entire stochastic search procedure is unlikely to yield credible results. In other words, the MC^3 algorithm could remain at an invalid model for many iterations and, quite possibly, never leave from it to visit other logistic regressions. You should know that, to the best of my knowledge, there does not exist any statistical software that will perform a similar type of model determination approach for logistic regressions. As such, the code you will write might turn up to be quite valuable for your applied projects that involve binary responses and many explanatory variables.

The MC^3 algorithm starts at a randomly generated logistic regression and proceeds for a large number of iterations. As opposed to the forward\backward greedy procedures you have already implemented, the MC^3 algorithm does not stop on its own: you will need to specify the number of iterations you want to run it for. How many iterations are needed by MC^3 to find the solution of the problem (1)? There is no clear answer to this question. In theory, MC^3 finds the solution of the problem (1) with probability 1 given that it is run for "enough" iterations. In practice, the only way to empirically verify you have run MC^3 long enough is to employ it starting with several randomly generated models and check if it has returned the same best model. This is what you are going to do in the second part of your assignment.

Problem 1

You are asked to implement in R the MC^3 algorithm. The procedure starts at a randomly generated logistic regression (iteration 0) and continues exploring the space of logistic regressions at iteration $r = 1, 2, ..., R$. You can assume you know the total number of iterations R, but your implementation should allow the user to run MC^3 for any number of iteration R of their choice.

The algorithm keeps track of two models: the current model $\mathcal{M}_{A^{(r)}}(A^{(r)} \subseteq V)$ and the best model $\mathcal{M}_{B^{(r)}}(B^{(r)} \subseteq V)$. The best model is the model with the smallest AIC that has been visited at the previous iterations, i.e.

$$AIC(\mathcal{M}_{B^{(r)}}) = \min\{AIC(\mathcal{M}_{A^{(r')}}) : r' = 0, 1, ..., r\},$$

or, equivalently,

$$AIC(\mathcal{M}_{B^{(r)}}) = \min\{AIC(\mathcal{M}_{B^{(r-1)}}), AIC(\mathcal{M}_{A^{(r)}})\}, \quad r \geq 1. \tag{2}$$

Remark that the forward\backward greedy algorithms from Homework 2 did not have to separately keep track of the best model and of the current model since the current model was always the best model identified so far.

Iteration 0: Generating a random starting logistic regression model.

You need to randomly generate a subset $A^{(0)}$ of $V = \{1, 2, ..., p\}$ such that $\mathcal{M}_{A^{(0)}}$ is a valid logistic regression whose MLEs exist (hence whose AIC score can be numerically trusted). To this end, you need to sample uniformly a number k from the set V. This represents the number of explanatory variables you will select. Next you sample without replacement k numbers from V. You will obtain a subset $A^{(0)}$ that contains k different elements of V. The R functions sample and sample.int could be useful for these tasks. Next you need to use the function isValidLogisticRCDD from my solution to Homework 2 to test whether $\mathcal{M}_{A^{(0)}}$ is a valid logistic regression. If the test fails, you must keep sampling subsets $A^{(0)}$ until you obtain one that corresponds with a logistic regression whose MLEs exist.
At the completion of Iteration 0 you need to set $B^{(0)} = A^{(0)}$ since $\mathcal{M}_{A^{(0)}}$ is the only model you visited so far.

Iteration r.

You need to follow these steps:

Step 1. Identify the neighbors of the current logistic regression $\mathcal{M}_{A^{(r)}}$. These neighbors are obtained by adding one variable to the set of explanatory variables $A^{(r)}$ or by deleting one variable from $A^{(r)}$, i.e.

$$\text{nbd}(\mathcal{M}_{A^{(r)}}) = \left(\bigcup_{i \in V \setminus A^{(r)}} \{A^{(r)} \cup \{i\}\} \right) \bigcup \left(\bigcup_{i \in A^{(r)}} \{A^{(r)} \setminus \{i\}\} \right).$$

Step 2. Use the function isValidLogisticRCDD to eliminate from $\text{nbd}(\mathcal{M}_{A^{(r)}})$ all the logistic regressions whose MLEs do not exist. We denote by $\text{nbd}^{valid}(\mathcal{M}_{A^{(r)}})$ the resulting set of valid neighbors of $\mathcal{M}_{A^{(r)}}$.

Step 3. Uniformly sample a model $\mathcal{M}_{A^{(r)}}$ from $\text{nbd}^{valid}(\mathcal{M}_{A^{(r)}})$. The function sample and sample.int could be useful here.

Step 4. Determine the set of valid neighbors of $\mathcal{M}_{A'}$, denoted by $\mathcal{M}_{A'}$. Here you repeat

Steps 1 and 2 above for $\mathcal{M}_{A'}$ instead of $\mathcal{M}_{A^{(r)}}$. If you developed nice functions for performing each step, you should not have to write any new code at Step 4.

Step 5. Calculate

$$p_{A'} = -AIC(\mathcal{M}_{A'}) - \log\left(\#\left(\text{nbd}^{valid}(\mathcal{M}_{A'})\right)\right).$$

Here $\#\left(\text{nbd}^{valid}(\mathcal{M}_{A'})\right)$ represents the number of elements (models) in the set of models $\text{nbd}^{valid}(\mathcal{M}_{A'})$.

Step 6. Calculate

$$p_{A^{(r)}} = -AIC(\mathcal{M}_{A^{(r)}}) - \log\left(\#\left(\text{nbd}^{valid}(\mathcal{M}_{A^{(r)}})\right)\right).$$

Step 7. If $p_{A'} > p_{A^{(r)}}$, $\mathcal{M}_{A'}$ becomes the current model. That is, set $A^{(r+1)} = A'$ and use equation (2) to find out the best logistic regression $\mathcal{M}_{B^{(r+1)}}$ visited so far. More explicitly, if $AIC(\mathcal{M}_{A'} < AIC(\mathcal{M}_{B^{(r)}}))$ set $B^{(r+1)} = A'$. Otherwise set $B^{r+1} = B^{(r)}$. Move to Step 9 (i.e., skip the next step).

Step 8. If $p_{A'} \leq p_{A^{(r)}}$, sample u from the uniform distribution on (0, 1). The R function runif could be useful. If $\log(u) < p_{A'} - p_{A^{(r)}}$, $\mathcal{M}_{A'}$ becomes the current model. Perform the same updates as in Step 7. If $\log(u) \geq p_{A'} - p_{A^{(r)}}$, $\mathcal{M}_{A^{(r)}}$ remains the current model (i.e., the Markov chain does not move to the proposed state $\mathcal{M}_{A'}$). In this case you set $A^{(r+1)} = A^{(r)}$ and $B^{(r+1)} = B^{(r)}$.

Step 9. If the current maximum number of iterations has been reached (i.e., if $r = R$), STOP. Otherwise continue to the next iteration.

The MC^3 algorithm should output the indices of the explanatory variables associated with the best logistic regression model identified as well as the value of the AIC of this best model. In other words, you should output $B^{(R)}$ and $AIC(\mathcal{M}_{B^{(R)}})$.

Answer:

My algorithm is like below,

```
1 isValidLogisticRCDD <- function(response,explanatory,data)
2 {
3   if(0==length(explanatory))
4   {
5     #we assume that the empty logistic regresion is valid
6     return(TRUE); # if condition satisfied, the function will
          stop here and give answer
7   }
```

```
8   # suppressWarnings: not show the warnings
9   logisticreg = suppressWarnings(glm(data[,response] ~ as.
      matrix(data[,as.numeric(explanatory)]),family=binomial(
      link=logit),x=TRUE));
10  tanv = logisticreg$x;
11  tanv[data[,response] == 1, ] <- (-tanv[data[,response] ==
      1, ]); # let the data that y actual value is one be minus
      ?
12  vrep = cbind(0, 0, tanv);  # add two column of 0 in front
      of tanv
13  #with exact arithmetic; takes a long time
14  #lout = linearity(d2q(vrep), rep = "V");
15
16  lout = linearity(vrep, rep = "V");
17  return(length(lout)==nrow(data));
18 }
19
20 findNeighbor <- function(response, explanatory, data, subset_
     a){
21  # find the neighbors
22  neighbor_lst <- vector(mode = "list", length = length(
      explanatory))
23  for (i in 1:length(explanatory)){
24    test_sub <- subset_a
25    if (explanatory[i] %in% test_sub){
26      neighbor_lst[[i]] <- setdiff(test_sub, explanatory[i])
27    } else {
28      neighbor_lst[[i]] <- append(test_sub, explanatory[i])
29    }
30  }
31  # eliminate the variables whose MLEs not exists
32  for (j in 1:length(neighbor_lst)){
33    if (isValidLogisticRCDD(response, neighbor_lst[[j]], data
       ) == FALSE){
34      #print("here we eliminate variables")
35      neighbor_lst[j] <- list(NULL)
36    }
37  }
38
39  # neighbor list valid
40  neighbor_lst_valid <- neighbor_lst[lengths(neighbor_lst) !=
      0]
41  return(neighbor_lst_valid)
42 }
43
44 MC3search <- function(response, data, n_iter = 3){
45  explanatory <- c(1:(response-1))
46
```

```
47  # iteration 0
48  k <- sample.int(length(explanatory), 1)
49  subset_a <- c(sample(explanatory, k))
50  while (isValidLogisticRCDD(response, subset_a, data) ==
        FALSE){
51    k <- sample.int(length(explanatory), 1)
52    subset_a <- c(sample(explanatory, k))
53  }
54  subset_b <- subset_a
55  cat("The start model is:          ", subset_b, "\n")
56
57  for (i in 1:n_iter){
58    cat("The", i, "iterations.\n")
59    # iteration r
60    neighbor_lst_valid <- findNeighbor(response, explanatory,
          data, subset_a)
61
62    # sample from valid neighbor
63    choose <- sample(length(neighbor_lst_valid), 1)
64    neighbor_lst_valid_prime <- findNeighbor(response,
        explanatory, data, neighbor_lst_valid[[choose]])
65
66    # step 5
67    model_prime <- glm(data[,response] ~ as.matrix(data[,as.
        numeric(c(neighbor_lst_valid[[choose]]))]),family=
        binomial(link=logit))
68    p_a_prime <- -model_prime$aic - log(length(neighbor_lst_
        valid_prime))
69
70    # step 6
71    model_r <- glm(data[,response] ~ as.matrix(data[,as.
        numeric(c(subset_a))]),family=binomial(link=logit))
72    p_a_r <- -model_r$aic - log(length(neighbor_lst_valid))
73
74    if (p_a_prime > p_a_r){
75      #print("Falls into first condition")
76      subset_a <- neighbor_lst_valid[[choose]]
77      model_a <- glm(data[,response] ~ as.matrix(data[,as.
          numeric(c(subset_a))]),family=binomial(link=logit))
78      model_b <- glm(data[,response] ~ as.matrix(data[,as.
          numeric(c(subset_b))]),family=binomial(link=logit))
79      #Mbr_aic <- min(model_b$aic , model_a$aic)
80      if (model_a$aic < model_b$aic){
81        #print("Falls into first_first condition")
82        subset_b <- subset_a
83      } else {
84        #print("Falls into first_second condition")
85        subset_b <- subset_b
```

```
      }
    } else if (p_a_prime <= p_a_r){
      #print("Falls into second condition")
      u <- runif(1, 0, 1)
      if (log(u) < (p_a_prime - p_a_r)){
        #print("Falls into second_first condition")
        subset_a <- neighbor_lst_valid[[choose]]
        model_a <- glm(data[,response] ~ as.matrix(data[,as.
           numeric(c(subset_a))]),family=binomial(link=logit))
        model_b <- glm(data[,response] ~ as.matrix(data[,as.
           numeric(c(subset_b))]),family=binomial(link=logit))
        #Mbr_aic <- min(model_b$aic, model_a$aic)
        if (model_a$aic < model_b$aic){
          #print("Falls into second_first_first condition")
          subset_b <- subset_a
        } else {
          #print("Falls into second_first_second condition")
          subset_b <- subset_b
        }
      }
    }
    cat("The iteration best model is: ", subset_b, "\n")
  }

  best_model <- glm(data[, response] ~ as.matrix(data[, as.
     numeric(c(subset_b))]), family = binomial(link = logit))
  return(list(bestAICvars = subset_b, bestAIC = best_model$
     aic))
}
```

I let the algorithm to print out the model for every iteration to make sure it goes right.

Problem 2

Run 10 instances of the MC^3 algorithm you implemented at Problem 1 for $R = 25$ iterations for the "534binarydata.txt" data. Comment on your findings.

Answer:

I run the whole program for two times, get the 20 AIC of the algorithm, and I found that all the result are around 60 to 95. Also, I found that the more variable we use in the model, our AIC will be bigger, but on the other hand, the less variables we use in the model, doesn't straight forwardly give us the smaller AIC. Besides, the number of the variables we choose in the model are mostly about 20, we don't have the model that its number of variable with

very big size. I'll put my result in a separate txt file for reference, the r code as well.