統計學習初論（105-2）

作業二

作業設計：盧信銘
國立台灣大學資管系

截止時間：2017 年 3 月 28 日上午 9 點（第二、三題）
2017 年 4 月 11 日上午 9 點 (第一題)

第一題請至 RSAND 上批改，範例命令：sl_check_hw2q1 ./your_program。第二題批改範例命令：sl_check_hw2q2 ./your_program。第三題改範例命令：sl_check_hw2q3 ./your_program。作業自己做。嚴禁抄襲。不接受紙本繳交，不接受遲交。請以英文或中文作答。

第一題

(50 points) Write a function named lm_evmax that takes the outcome matrix *y* and a feature matrix *xmat*, and perform evidence maximization. The matrix *y* should be a matrix of one column, and the first column of the *xmat* matrix should be all ones. Follow the following procedure closely to implement lm_evmax. We should start with an initial estimation that is not too far from the optimal solution. To do this, use the regularized least square solution in Section 3.1.14. Set $\lambda = 0.001\ N$, where N is the number of observation in $xmat$. We can now estimate the initial value for m_N by $w = (\lambda I + xmat^T xmat) xmat^T y$. The initial value for β is: $\frac{1}{\beta} = \frac{N}{e_0^T e_0}$, where $e_0 = y - xmat\ w$. The initial α, following our discussion in class, is $\alpha = \lambda\ \beta$.

With all the initial values, we can now start to iterate by computing

$A = \alpha I + \beta xmat^T xmat,$

$m_N = \beta A^{-1} xmat^T y,$

$\gamma = \sum_{i=1}^{M} \frac{\lambda_i}{\alpha + \lambda_i}.$

The new α is $\alpha_{new} = \frac{\gamma}{m_N^T m_N}$, and the new β is $\frac{1}{\beta_{new}} = \frac{e_1^T e_1}{N - \gamma}$, where $e_1 = y - xmat\ m_N$.

Iterate until the $(|\alpha - \alpha_{new}| + |\beta - \beta_{new}|) < 10^{-5}$.

Your function should output a list that store m_N under the name of mN, the square root of the diagonal elements of A under the name of mNsd, α under the name of alpha, and β under the name of beta.

Sample input and output:

```
> setwd('your_path_to_data')
> load(file='rtb2_train.rdata')
>
> nfeat=20
> rtb3 = rtb2_train[1:(nfeat+1)]
> y=as.matrix(rtb3[,1])
> xmat = model.matrix(paying_price~., data=rtb3)
> lmev1 = lm_evmax(y, xmat)
> lmev1
$mN
                  [,1]
(Intercept)  98.940003
agent_1      43.452779
agent_2     114.194678
agent_3       6.546850
agent_4     -15.585023
agent_5       1.861286
agent_6       6.272122
agent_7      17.879145
agent_8       5.386818
agent_9     -12.751464
agent_10     -3.897753
agent_11     -7.807724
agent_12     -2.450378
agent_13     -4.953377
agent_14      5.720329
agent_15      8.330334
agent_16      9.683872
agent_17      1.678195
agent_18     -5.640699
agent_19     -9.099482
agent_20     -7.276850

$mNsd
(Intercept)     agent_1     agent_2     agent_3     agent_4     agent_5
  3.0069519   5.6476653   4.3677002   5.3702176   3.7473316   3.2040479
    agent_6     agent_7     agent_8     agent_9    agent_10    agent_11
  7.7447867   7.5544160   6.3807694   3.9323111   0.6219326   1.0473609
   agent_12    agent_13    agent_14    agent_15    agent_16    agent_17
  2.9611644  24.2577922  24.7049856  11.4733189   5.3802582   3.8772171
   agent_18    agent_19    agent_20
  3.6513938   2.4602313   3.0162630

$alpha
[1] 0.0007585334

$beta
[1] 0.0002535907
```

Evaluation: All credits will be given based on the correctness of 10 testing cases. Correct output in a case is worth 5 points.

第二題

(25 points) Write a function named gen_utagmat to generate a matrix that contains dummy coding of the user_tags column in rtb1_train.rdata. The gen_utagmat function takes two arguments. The first argument, utagvec, contains the column of strings of comma separated user tags. The second argument, y, contains the column of paying_price. Follow the following instruction to process the data.

1. For each row in utagvec, split the user tags string by comma (“,”).
2. Count the frequency of each user tag, and remove user tags that appeared less than five times.
3. Use simple regression to compute the t-value for each user tag.
4. Remove user tags with an absolute value of t-value less than one.
5. Order the remaining user tags by the absolute value of t-value (from large to small).
6. Generate the output matrix. There should be 1+p columns in this matrix. Store the user tag dummy by the order from the previous step.
7. Add names to the columns of the output matrix, the first column is named “constant.” The remaining columns should be named as “user_???,” where ??? are the user tag string. For example, for user tag 16706, its column names is user_16706.
8. Return the matrix constructed in the previous step.

To save your time, I listed a few key functions that maybe useful for you:

- strsplit: a function that can split a column of strings by a character.
- table: count user tag frequency.
- sapply: can be used to apply an operation (defined by a function) to every element in a column.
- %in%: an operator to check whether an element is present in a data structure.

Sample input and output:

```
> setwd('your_path_to_data')
> load(file='rtb1_train.rdata')
> rtb1_train = rtb1_train[1:300,]
> umat1 = gen_utagmat(rtb1_train$user_tags,
rtb1_train$paying_price)
> head(umat1)
     constant user_10063 user_10111 user_10006 user_10077 user_14273 user_10059
[1,]        1          1          1          0          0          0          0
[2,]        1          1          0          0          0          0          0
[3,]        1          1          1          1          0          0          0
[4,]        1          1          0          1          0          0          0
[5,]        1          1          0          1          0          0          0
[6,]        1          0          0          1          0          0          0
     user_10057 user_13776 user_13800 user_10052 user_10079 user_13678
[1,]          0          0          0          0          0          0
```

```
[2,]          0          0          0          0          0          0
[3,]          1          0          1          0          0          0
[4,]          0          0          0          0          0          0
[5,]          0          0          1          0          0          0
[6,]          1          0          0          1          0          0
> y = rtb1_train$paying_price
> w = solve(t(umat1) %*% umat1, t(umat1) %*% y)
> print(w)
                 [,1]
constant    99.038607
user_10063  -8.884685
user_10111  -8.799026
user_10006  -8.181896
user_10077 -10.550929
user_14273  30.256580
user_10059  -2.541225
user_10057  -7.892633
user_13776  19.140186
user_13800  -5.505180
user_10052 -13.681258
user_10079  27.587216
user_13678 -28.505499
```

Evaluation: All credits will be given based on the correctness of 10 testing cases. Correct output in a case is worth 2.5 points.

第三題

(25 points) Similar to the previous question, write a function named gen_uagentmat to generate a matrix that contains dummy coding of the user_agent column in rtb1_train.rdata. The gen_ uagentmat function takes two arguments. The first argument, uagentvec, contains the column of strings of user agents. The second argument, y, contains the column of paying_price. The user_agent column looks like this:

```
> head(rtb1_train$user_agent)
[1] "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/535.12 (KHTML, like Gecko) Maxthon/3.0
Chrome/18.0.966.0 Safari/535.12"
[2] "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)"
[3] "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 (KHTML, like Gecko)
Chrome/21.0.1180.89 Safari/537.1"
[4] "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322)"
[5] "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 (KHTML, like Gecko)
Chrome/21.0.1180.89 Safari/537.1"
[6] "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)"
```

To simplify the problem, we are going to extract “word-like” strings for subsequent analysis. For example, we are going to extract the following words from the first row listed above: Mozilla, Windows, NT, AppleWebKit, KHTML, like, Gecko, Maxthon, Chrome, Safari. This can be done by following code segment:

```
> #define the input vector
> utagstr=c("Mozilla/5.0 (Windows NT 5.1) AppleWebKit/535.12 (KHTML,
like Gecko) Maxthon/3.0 Chrome/18.0.966.0 Safari/535.12",
+   "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)",
```

```
+   "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 (KHTML,
like Gecko) Chrome/21.0.1180.89 Safari/537.1")
> #define regular expression pattern
> pattern <- "([A-Za-z][A-Za-z0-9]{1,})"
> #do regular expression matching.
> list2=regmatches(utagstr, gregexpr(pattern, utagstr))
> #keep only unique words in each row.
> list2=lapply(list2, unique)
> list2
[[1]]
 [1] "Mozilla"     "Windows"     "NT"          "AppleWebKit" "KHTML"
"like"        "Gecko"       "Maxthon"     "Chrome"
[10] "Safari"

[[2]]
[1] "Mozilla"    "compatible" "MSIE"       "Windows"    "NT"
"SV1"

[[3]]
 [1] "Mozilla"     "Windows"     "NT"          "WOW64"
"AppleWebKit" "KHTML"       "like"        "Gecko"       "Chrome"
[10] "Safari"
```

The regular expression pattern `pattern <- "([A-Za-z][A-Za-z0-9]{1,})"` will match words start with an letter, but allow the word ends with a digit (e.g., SE1). You should study the regular expression document if you are not familiar with the syntax.

The remaining steps are similar to those outlined in the previous question. This function should return the matrix that contains ones in the first column, and dummy coding of keywords in subsequent columns ordered by the absolute t-value (from large to small).

We are going to apply different frequency thresholds in this question. Define document frequency of a feature as the number of rows that contain the word. We are going to include features with (1) a document frequency equal or larger than 10, and (2) a document frequency less than or equal to floor(0.5N), where N is the total number of input data points. Note that you can use the “unique” function to remove duplicated words in a record.

Sample input and output:

```
> setwd('your_path_to_data')
> load(file='rtb1_train.rdata')
> rtb1_train = rtb1_train[1:1500,]
> y = rtb1_train$paying_price
> umat1 = gen_uagentmat(rtb1_train$user_agent,y)
>
> print(head(umat1))
     constant agent BIDUPlayerBrowser agent Trident agent Version agent MALN
[1,]        1                       0             0             0          0
[2,]        1                       0             0             0          0
[3,]        1                       0             0             0          0
[4,]        1                       0             1             0          0
[5,]        1                       0             0             0          0
[6,]        1                       0             1             0          0
```

```
     agent_Mobile agent_QQBrowser agent_qdesk agent_rv agent_zh agent_NET4
[1,]            0               0           0        0        0          0
[2,]            0               0           0        0        0          0
[3,]            0               0           0        0        0          0
[4,]            0               0           0        0        0          0
[5,]            0               0           0        0        0          0
[6,]            0               0           0        0        0          0
     agent_MetaSr agent_SE agent_LBBROWSER agent_Android agent_SV1 agent_Build
[1,]            0        0               0             0         0           0
[2,]            0        0               0             0         1           0
[3,]            0        0               0             0         0           0
[4,]            0        0               0             0         0           0
[5,]            0        0               0             0         0           0
[6,]            0        0               0             0         0           0
     agent_cn agent_CIBA agent_NET agent_CLR agent_SLCC2 agent_OS agent_Mac
[1,]        0          0         0         0           0        0         0
[2,]        0          0         0         0           0        0         0
[3,]        0          0         0         0           0        0         0
[4,]        0          0         1         1           0        0         0
[5,]        0          0         0         0           0        0         0
[6,]        0          0         0         0           0        0         0
     agent_Maxthon agent_Linux
[1,]             1           0
[2,]             0           0
[3,]             0           0
[4,]             0           0
[5,]             0           0
[6,]             0           0
> print(head(sort(colSums(umat1), decreasing=TRUE), n=10))
     constant agent_Trident     agent_NET     agent_CLR     agent_SV1
         1500           707           416           416           228
     agent_SE   agent_SLCC2  agent_MetaSr    agent_NET4  agent_Mobile
          125           117           112            92            41
>
> #remove linearly independent columns
> qr1 = qr(umat1, tol =1e-7)
> ind3 = qr1$pivot[1:qr1$rank]
> rank0 = ncol(umat1)
> if(qr1$rank < rank0) {
+     cat("There are", rank0, "columns, but rank is only", qr1$rank,
"\n")
+     toremove = qr1$pivot[(qr1$rank+1):rank0]
+     cat("list of features removed", toremove,"\n")
+     tokeep = qr1$pivot[1:qr1$rank]
+     umat1 = umat1[,tokeep]
+ }
There are 26 columns, but rank is only 24
list of features removed 21 24
>
>
> w = solve(t(umat1) %*% umat1, t(umat1) %*% y)
> print(w)
                              [,1]
constant                85.9165566
agent_BIDUPlayerBrowser 50.7650527
agent_Trident            0.3618706
agent_Version            6.4907247
agent_MALN             -24.0530691
agent_Mobile            -0.4130522
agent_QQBrowser        -14.0837423
agent_qdesk            -16.8323664
agent_rv                35.2456083
```

```
agent_zh                -22.6896560
agent_NET4               -3.4547497
agent_MetaSr             -4.3156216
agent_SE                 -1.3502527
agent_LBBROWSER          16.5067009
agent_Android            48.2981819
agent_SV1                 8.1623468
agent_Build              19.8848795
agent_cn                 15.8460432
agent_CIBA               27.7160645
agent_NET                -2.7181741
agent_SLCC2               2.6677004
agent_OS                 10.4349814
agent_Maxthon           -12.2153875
agent_Linux             -36.4567017
```

Evaluation: All credits will be given based on the correctness of 10 testing cases. Correct output in a case is worth 2.5 points.