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: $\beta = \frac{N}{e_0^T e_0}$, 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^{-1} under the name of mNsd, α under the name of alpha, and β under the name of beta.

Sample input and output 1:

```
> 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)  98.941001
agent_1      43.453947
agent_2     114.196607
agent_3       6.545915
agent_4     -15.585473
agent_5       1.861077
agent_6       6.271456
agent_7      17.880470
agent_8       5.386568
agent_9     -12.751414
agent_10     -3.897748
agent_11     -7.807772
agent_12     -2.450647
agent_13     -4.958611
agent_14      5.725306
agent_15      8.330833
agent_16      9.684011
agent_17      1.678518
agent_18     -5.641692
agent_19     -9.100067
agent_20     -7.277845

$mNsd
(Intercept)     agent_1     agent_2     agent_3     agent_4
  3.0052509   5.6445189   4.3651917   5.3672246   3.7451977
    agent_5     agent_6     agent_7     agent_8     agent_9
  3.2022098   7.7406664   7.5503850   6.3772277   3.9300660
   agent_10    agent_11    agent_12    agent_13    agent_14
  0.6215702   1.0467525   2.9594740  24.2552267  24.7024972
   agent_15    agent_16    agent_17    agent_18    agent_19
 11.4676217   5.3772346   3.8749929   3.6493229   2.4588253
   agent_20
  3.0145566

$alpha
[1] 0.0007585334

$beta
[1] 0.0002538865
```

Sample input and output 2:

```
> setwd('your_path_to_data')
> load(file='rtb2_train.rdata')
>
> nfeat=seq(1, length(rtb2_train), by = 50)
> rtb3 = rtb2_train[1:10000,nfeat]
> y=as.matrix(rtb3[,1])
> xmat = model.matrix(paying_price~., data=rtb3)
> lmev1 = lm_evmax(y, xmat)
> lmev1
$mN
                     [,1]
(Intercept)    100.4745627
agent_50        24.0762994
agent_103       10.1641802
agent_158        4.0172520
user_10057      -8.2734590
url_16          93.3343427
url_66          81.9263975
url_116         31.3215269
url_166         -0.8606502
url_216        -34.1168234
url_266         -2.4165514
url_316         -9.4242748
url_366          0.0000000
url_416        -39.4635232
url_466         12.1393132
url_516          0.0000000
url_566        -51.2127917
url_616         38.1233404
url_666         15.6048899
url_716          0.0000000
url_766        -25.9820298
url_816        -12.8898514
url_866          0.0000000
url_916         13.3998047
url_966          0.0000000
url_1016         0.0000000
url_1066        17.9632028
url_1116         0.0000000
ad_slot_id_1   -41.9625073
ad_slot_id_51   13.9047828
ad_slot_id_101 -18.7191215
ad_slot_id_151  48.2599954
ad_slot_id_201  51.0284580
ad_slot_id_251  36.3351298
ad_slot_id_301 -26.3513146
ad_slot_id_351  49.0202346
ad_slot_id_401   7.1334803
ad_slot_id_451  41.0191186
ad_slot_id_501   1.1261907
ad_slot_id_551   0.0000000
ad_slot_id_601  11.9589780
ad_slot_id_651  14.1927424
ad_slot_id_701   0.0000000
ad_slot_id_751  -6.0329717
ad_slot_id_801   0.0000000
domain_46       94.1003037
domain_96        0.0000000
domain_146     -21.9265105
domain_196       0.0000000
domain_246     -35.7886961
```

```
domain_296      37.1806638
domain_346      12.5243773
domain_396       0.0000000
domain_446       0.0000000
domain_496     -45.8577932
domain_546      23.9770084
domain_596      -6.0329717

$mNsd
   (Intercept)        agent_50       agent_103       agent_158      user_10057
     0.7379481     36.9458157     15.2760156     14.2058923      1.5374625
        url_16         url_66        url_116        url_166        url_216
    10.0060958     20.3319885     23.1867582     15.6205344     17.4960124
       url_266        url_316        url_366        url_416        url_466
    20.3139447     36.9458157     47.1126442     31.3945727     36.9458157
       url_516        url_566        url_616        url_666        url_716
    47.1126442     27.7725249     25.1708054     36.9458157     47.1126442
       url_766        url_816        url_866        url_916        url_966
    36.9458157     36.9458157     47.1126442     36.9490225     47.1126442
      url_1016       url_1066       url_1116   ad_slot_id_1  ad_slot_id_51
    47.1126442     27.7725249     47.1126442      1.4056166      9.3709884
ad_slot_id_101 ad_slot_id_151 ad_slot_id_201 ad_slot_id_251 ad_slot_id_301
     8.4810507     27.7725819     27.7725249     37.4095318     16.7858621
ad_slot_id_351 ad_slot_id_401 ad_slot_id_451 ad_slot_id_501 ad_slot_id_551
    31.3955976     36.9458157     36.9458157     31.3945727     47.1126442
ad_slot_id_601 ad_slot_id_651 ad_slot_id_701 ad_slot_id_751 ad_slot_id_801
    18.2999500     31.3945727     47.1126442     35.9801340     47.1126442
     domain_46      domain_96     domain_146     domain_196     domain_246
    27.7725249     47.1126442     20.3160959     47.1126442     14.2068564
    domain_296     domain_346     domain_396     domain_446     domain_496
    18.2988486     36.9458157     47.1126442     47.1126442     31.3945727
    domain_546     domain_596
    21.6084555     35.9801340

$alpha
[1] 0.0004505314

$beta
[1] 0.0002821163
```

Sample input and output 3:

```
> setwd('your_path_to_data')
> load(file='rtb2_train.rdata')
>
> nfeat=seq(1, length(rtb2_train), by = 40)
> rtb3 = rtb2_train[nfeat]
> y=as.matrix(rtb3[,1])
> xmat = model.matrix(paying_price~., data=rtb3)
> lmev1 = lm_evmax(y, xmat)
> lmev1
$mN
                     [,1]
(Intercept)    101.8302808
agent_40         5.5742646
agent_83         5.5949491
agent_125       10.6679739
agent_169       10.3619323
user_10057      -8.1791695
url_6          -66.2237601
url_46           1.5576434
url_86           0.2777827
url_126        -41.9386318
url_166          0.5658397
url_206         76.2007520
url_246        -63.4182569
url_286         -2.1062712
url_326        -66.8628019
url_366        -50.9903056
url_406         21.0375925
url_446         23.7072943
url_486        -53.5287665
url_526        -54.9506555
url_566        -35.3568148
url_606         29.4711094
url_646         21.9153860
url_686         26.7922792
url_726        -39.7309547
url_766        -41.2635100
url_806         16.6446985
url_846         -0.9691111
url_886        -37.7676340
url_926         13.2095923
url_966         -0.4014126
url_1006        11.8079327
url_1046        -1.1496182
url_1086        16.3418655
url_1126       -25.9745522
ad_slot_id_1   -43.5619818
ad_slot_id_41   53.8200654
ad_slot_id_81   26.2876093
ad_slot_id_121  24.4952044
ad_slot_id_161  45.3289173
ad_slot_id_201  63.1118155
ad_slot_id_241  43.7379560
ad_slot_id_281 -67.4577704
ad_slot_id_321  18.5629974
ad_slot_id_361 -53.4090614
ad_slot_id_401  39.0564170
ad_slot_id_441  33.9566347
ad_slot_id_481  32.2257312
ad_slot_id_521  33.7058323
ad_slot_id_561  28.1896232
```

```
ad_slot_id_601   4.2483136
ad_slot_id_641  12.1009326
ad_slot_id_681 -15.3877926
ad_slot_id_721 -31.5576145
ad_slot_id_761 -26.1128588
ad_slot_id_801  12.8207545
domain_36       34.3964054
domain_76       76.3594959
domain_116      45.1899791
domain_156      11.7679577
domain_196      70.1342099
domain_236      59.9155922
domain_276      49.2458507
domain_316      52.1008217
domain_356      36.4211597
domain_396      24.2672065
domain_436      28.5159067
domain_476      21.2111630
domain_516      24.9081412
domain_556       1.9285828
domain_596       4.5992392

$mNsd
   (Intercept)        agent_40        agent_83       agent_125       agent_169
     0.2209299       2.4069776       4.5390369       8.6546355      13.7431687
    user_10057           url_6          url_46          url_86         url_126
     0.4512137       2.2399299       2.8632804       4.3313212       4.9832576
       url_166         url_206         url_246         url_286         url_326
     6.5114389      22.5081167      14.5044244       9.3616472      19.8816980
       url_366         url_406         url_446         url_486         url_526
    14.0979968      10.9340741      13.0577709      18.8719809      22.5081819
       url_566         url_606         url_646         url_686         url_726
    11.5266057      19.8817496      15.4348582      21.0731725      18.8719003
       url_766         url_806         url_846         url_886         url_926
    22.5081167      16.5709957      18.8734339      22.5081819      16.5709957
       url_966        url_1006        url_1046        url_1086        url_1126
    21.0743198      17.2420457      22.5091551      22.5081167      13.7242217
  ad_slot_id_1   ad_slot_id_41   ad_slot_id_81  ad_slot_id_121  ad_slot_id_161
     0.4248913      10.5791740       3.6283353       4.8996879       9.9839483
ad_slot_id_201  ad_slot_id_241  ad_slot_id_281  ad_slot_id_321  ad_slot_id_361
    15.4354625      12.4800296      15.4348138       8.1562509      13.7242217
ad_slot_id_401  ad_slot_id_441  ad_slot_id_481  ad_slot_id_521  ad_slot_id_561
    18.0019511      17.2420457      18.0018544      21.0731725      18.8719003
ad_slot_id_601  ad_slot_id_641  ad_slot_id_681  ad_slot_id_721  ad_slot_id_761
     7.7390422      13.7433142       2.3760928      15.9727699      13.7240643
ad_slot_id_801       domain_36       domain_76      domain_116      domain_156
    22.5081167      12.1422525      10.4297786       8.7433101       4.2342114
    domain_196      domain_236      domain_276      domain_316      domain_356
    18.8719809      18.0019363      17.2420457      21.0731725      17.2421446
    domain_396      domain_436      domain_476      domain_516      domain_556
    13.3783913      18.0019511      16.5710958      22.5081819       8.5612977
    domain_596
    11.9723083

$alpha
[1] 0.0005838557

$beta
[1] 0.0002780223
```

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) and feature name (reverse alphabetical order; if abs(t-value) is the same).

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.