統計學習初論（105-2）

作業一

作業設計：盧信銘
國立台灣大學資管系

截止時間：2017 年 2 月?日下午 1 點

第一題請至 RSAND 上批改，範例命令：sl_check_hw1q1 ./your_program。第二題的第一小題請上傳至 Ceiba 作業區。第二題第二小題請至 RSAND 上批改，範例命令：sl_check_hw1q2 ./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

(40 points) We are going to construct prediction models using the conditional distributions of multivariate Gaussian. Recall that for a random vector $x = [x_1\ x_2 \cdots x_D]$ that follows a multivariate Gaussian distribution with mean μ and covariance Σ. If we partition the random vector x into two groups, $x = \begin{bmatrix} x_a \\ x_b \end{bmatrix}$; $x_a = [x_1\ x_2\ \ldots x_M]^T$, and $x_b = [x_{M+1}\ \ldots x_D]^T$, then $p(x_a|x_b) = MN(\mu_{a|b}, \Sigma_{a|b})$, where

$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}$$

$$\mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b)$$

We are going to consider a special case that x_a only consists of one variable using a dataset that consists of 44 variables. This dataset are collected from a social media platform. The goal is to understand how a post on a company fan page reach the consumers. The first variable, life_post_consumer, is the number of people who clicked anywhere in the post. We want to construct a model that can predict this variable using the value of other variables. Thus, this variable is x_a and the remaining variables are x_b. The meaning of these variables are briefly described below.

Variable	Description
life_post_consumer	The number of people who clicked anywhere in the post.
comp_page_like	The number of likes on the company's fan page.
Paid	If the company paid to Facebook for advertising (1=yes).
life_post_reach	The number of people who saw a page post (unique users).

life_post_impression_liked	Total number of impressions just from people who have liked a page.
life_post_reach_liked	The number of people who saw a page post because they have liked that page (unique users).
comment	Number of comments on the publication.
like	Number of "Likes" on the publication.
share	Number of times the publication was shared.
type_link	Type of content is link sharing.
type_status	Type of content is status updates.
type_video	Type of content is video sharing. Note: photo sharing is represented as type_link=0 and type_status=0 and type_video=0.
cat2	Type of content is product (direct advertisement, explicit brand content).
cat3	Type of content is inspiration (non-explicit brand related content). Note: action (special offers and contests) is represented as cat2=0 and cat3=0.
month1 to month11	Posting month is Jan., Feb.,…, Nov. Note: Dec. is represented as month1=month2=…=month11=0
dow1 to dow6	Day of week is Sunday, Monday, …, Friday.
hour2 to hour14	Posting hour is 2 to 14.

Write a function named gpredict that takes a training data frame (dftrain) and an optional testing data frame (dftest). You should assume that the first column of dftrain is x_a while the remaining columns are x_b. The testing data frame (dftest) should contain the values of x_b only. You can assume that the dftrain and dftest is compatable. That is, the variables in dftrain and dftest have the same columns and are of the same order except for the first column in dftrain. Use dftrain to compute $\hat{\mu}_a$, $\hat{\mu}_b$, $\hat{\Sigma}_{ab}$, and $\hat{\Sigma}_{bb}$. Let x_i denote the i-th row in dftest, compute its prediction via $\hat{\mu}_a + \hat{\Sigma}_{ab}\hat{\Sigma}_{bb}^{-1}(x_i - \hat{\mu}_b)$. Your function should output a list that contains the following components: mua, mub, s_ab, s_bb, predict. The first component, mua, stores the value of $\hat{\mu}_a$, mub stores the value of μ_b, s_ab stores the value of $\hat{\Sigma}\_ab$, s_bb stores the value of $\hat{\Sigma}_{bb}$, and predict is a vector that contains the prediction of the testing data frame. The predict component should have a NULL value if dftest is not provided. A sample code segment that constructs the returned list (with all values set to zero) is as follows:

```
nfeature = ncol(dftest)
ntest = nrow(dftest)
ret=list(mua=0, mub=rep(0,nfeature),
         s_ab = matrix(0, nrow=1, ncol=nfeature),
         s_bb=matrix(0, nrow=nfeature, ncol=nfeature),
         predict=rep(0, ntest))
return(ret)
```

Sample input and output:

```
> options(scipen=10)
> df1_train = read.csv('df1_train.csv')
> df1_test1 = read.csv('df1_test1.csv')
> df1_test1y = read.csv('df1_test1y.csv')
>
> out1 = gpredict(df1_train[1:200,], df1_test1)
> print(out1$mua)
[1] 750.69
> print(out1$mub[1:5])
            comp_page_like                         Paid              life_post_reach
               123093.660                        0.275                    13268.630
life_post_impression_liked        life_post_reach_liked
                16135.520                     6366.485
> print(out1$s_ab[1:5])
            comp_page_like                         Paid              life_post_reach
            -1538198.40241                     18.67362                6314389.97518
life_post_impression_liked        life_post_reach_liked
            15093184.46352                   3128693.27171
> print(out1$s_bb[1:5,1:5])
                           comp_page_like         Paid life_post_reach
comp_page_like              267002163.9542  360.1894472     6302704.080
Paid                             360.1894    0.2003769        1205.389
life_post_reach               6302704.0796 1205.3886935   411470987.862
life_post_impression_liked -108781263.7469  694.1829146   392956330.480
life_post_reach_liked        -4833532.2564  440.9664573   108363037.427
                           life_post_impression_liked life_post_reach_liked
comp_page_like                        -108781263.7469         -4833532.2564
Paid                                         694.1829              440.9665
life_post_reach                        392956330.4798       108363037.4266
life_post_impression_liked            2349881335.3363       234692522.0581
life_post_reach_liked                  234692522.0581        52188995.7887
>
> mae1a = mean(abs(df1_test1y[,1] - out1$pred))
> cat("MAE1a=", mae1a, "\n")
MAE1a= 277.9231
```

Evaluation: All credits will be given based on the correctness of 10 testing cases. Correct output in a case is worth 4 points.

第二題

(60 points) We are going to look at the issue of sequential estimation in this problem. You can also find relevant discussion in Section 2.3.5 of PRML. Consider a set of observations $D = \{x_1, x_2, \dots, x_{N-1}\}$. Each x_i is a vector of length k, $k \geq 1$. If D is a sample from multivariate Gaussian, then we know that the MLE of mean and variance (i.e., covariance matrix) is:

$$\mu_{ML}^{N-1} = \frac{1}{N-1} \sum_{i=1}^{N-1} x_i \text{ and } \Sigma_{ML}^{N-1} = \frac{1}{N-1} \sum_{i=1}^{N-1} (x_i - \mu_{ML}^{N-1})(x_i - \mu_{ML}^{N-1})^T.$$

When we add one additional observation x_N to D, then the MLE of mean becomes:

$$\mu_{ML}^{N} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{1}{N}x_N + \frac{1}{N}\sum_{i=1}^{N-1} x_i = \frac{1}{N}x_N + \frac{N-1}{N}\mu_{ML}^{N-1}$$

$$= \mu_{ML}^{N-1} + \frac{1}{N}(x_N - \mu_{ML}^{N-1})$$

Thus, we do not need to go through all N observations again in order to compute the new MLE estimator of μ. Instead, we only need to use the update formula equation above to compute the new estimator for mean. This type of updating formula is quite useful if we are handling a very large dataset, and going through the whole dataset again is very time consuming. We are going to derive a similar updating formula for the covariance matrix, and implement a R function that can perform the task of computing the updated mean and covariance.

(1) (20 points) Derive the update formula that can move from Σ_{MLE}^{N-1} to Σ_{MLE}^{N}. This formula should take only μ_{MLE}^{N-1}, μ_{ML}^{N}, x_N, Σ_{MLE}^{N-1}, and N as inputs.

(2) (40 points) Write a R function named mle_update that takes three parameters:
mu: the original MLE estimator of μ,
s: the original MLE estimator of Σ,
n: number of observations in the original dataset,
x: the new observation.

Apply your updating formula and compute the update MLE for μ and Σ. Your function should return a list that contains three components: mu, s, and n. The three components should contain the values of the corresponding updated estimators.

Sample input and output:

```
> set.seed(1223)
> nobs = 3
> nfeature = 7
> rawdata=matrix(runif(nobs*nfeature), nrow=nobs, ncol=nfeature)
> data1 = rawdata[1:(nobs-1),]
> xn = rawdata[nobs,]
> cov1 = cov(data1)*(nrow(data1)-1) / nrow(data1)
> mu1 = colMeans(data1)
>
> out1 = mle_update(mu1, cov1, nrow(data1), xn)
> print(out1$mu[1:3])
[1] 0.3614303 0.4386873 0.5753175
> print(out1$s[1:3,1:3])
             [,1]        [,2]        [,3]
[1,]  0.042576147 0.001314261 -0.05248834
[2,]  0.001314261 0.081681814  0.04338238
[3,] -0.052488343 0.043382384  0.08951474
> print(out1$n)
[1] 3
```

Evaluation: All credits will be given based on the correctness of 10 testing cases. Correct output in a case is worth 4 points.