MAST90104: A First Course in Statistical Learning

Week 4 Workshop and Lab **Solutions**

Workshop questions

1. Suppose that X is a random variable with density function, f, given by

$$f(x) = \sum_{i=0}^{\infty} p(i)g(x;i)$$

where $p(0), p(1), \cdots$ is a discrete probability mass function on $\{0, 1, \cdots\}$ and each $g(x;i)$ is a probability density function. Suppose that $\mu(i), \sigma^2(i), M(t;i)$ are the mean, variance and moment generating function for the density $g(x;i)$. Let $M(t)$ be the moment generating function of X. Suppose also that N is a random variable with probability mass function $p(i), i = 0, 1, \cdots$. Show that

(a) $E(X) = E(\mu(N))$

Solution:

$$\begin{aligned}
E(X) &= \int_{-\infty}^{\infty} x f(x)\, dx \\
&= \int_{-\infty}^{\infty} x \sum_{i=0}^{\infty} p(i)g(x;i)\, dx \\
&= \sum_{i=0}^{\infty} \int_{-\infty}^{\infty} x g(x;i)\, dx p(i) \\
&= \sum_{i=0}^{\infty} \mu(i)p(i) \\
&= E(\mu(N))
\end{aligned}$$

(b) $\text{var}\ (X) = E(\sigma^2(N)) + \text{var}\ (\mu(N))$

Solution:

$$\begin{aligned}
\text{var}\ (X) &= E(X^2) - (E(X))^2 \\
&= \int_{-\infty}^{\infty} x^2 \sum_{i=0}^{\infty} p(i)g(x;i)\, dx - (E(X))^2 \\
&= \sum_{i=0}^{\infty} \int_{-\infty}^{\infty} x^2 g(x;i)\, dx p(i) - (E(X))^2 \\
&= \sum_{i=0}^{\infty} (\sigma^2(i) + \mu^2(i))p(i) - (E(X))^2 \\
&= E(\sigma^2(N)) + E(\mu^2(N)) - E(\mu(N)))^2 \\
&= E(\sigma^2(N)) + \text{var}\ (\mu(N))
\end{aligned}$$

(c) $M(t) = E(M(t;N))$.

Solution:

$$\begin{aligned}
M(t) &= E(e^{tX}) \\
&= \int_{-\infty}^{\infty} e^{tx} \sum_{i=0}^{\infty} p(i) g(x;i)\, dx \\
&= \sum_{i=0}^{\infty} \int_{-\infty}^{\infty} e^{tx} g(x;i)\, dx p(i) \\
&= \sum_{i=0}^{\infty} M(t;i) p(i) \\
&= E(M(t;N))
\end{aligned}$$

(Hint: You may assume that interchange of infinite sums and integrals is justified.)

2. Let $y_1, \ldots, y_n$ be an i.i.d. normal sample. Show that

$$\bar{y} = \tfrac{1}{n} \sum_{i=1}^{n} y_i \text{ and } s^2 = \tfrac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2$$

are independent. (*Hint*: Express them as a random "vector" and quadratic form respectively.)

Solution: Let $\mathbf{1}$ be the vector made up entirely of 1's, then

$$\begin{aligned}
\bar{y} &= \tfrac{1}{n}\mathbf{1}^T\mathbf{y} \\
\mathbf{y} - \bar{y}\mathbf{1} &= (I - \tfrac{1}{n}\mathbf{1}\mathbf{1}^T)\mathbf{y} \\
s^2 &= \tfrac{1}{n-1}[(I - \tfrac{1}{n}\mathbf{1}\mathbf{1}^T)\mathbf{y}]^T (I - \tfrac{1}{n}\mathbf{1}\mathbf{1}^T)\mathbf{y} \\
&= \tfrac{1}{n-1}\mathbf{y}^T (I - \tfrac{1}{n}\mathbf{1}\mathbf{1}^T)^T (I - \tfrac{1}{n}\mathbf{1}\mathbf{1}^T)\mathbf{y} \\
&= \tfrac{1}{n-1}\mathbf{y}^T (I - \tfrac{1}{n}\mathbf{1}\mathbf{1}^T)\mathbf{y}
\end{aligned}$$

noting that $I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ is symmetric and idempotent. It is now easy to check that $B = \frac{1}{n}\mathbf{1}^T$ and $A = \frac{1}{n-1}(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)$ satisfy $BVA = 0$, where $V = \sigma^2 I = \operatorname{Var}\mathbf{y}$, whence $\bar{y} = B\mathbf{y}$ and $s^2 = \mathbf{y}^T A\mathbf{y}$ are independent.

The alternative approach is to notice that $\bar{y}$ and s^2 are just the usual estimates of $\boldsymbol{\beta}$ and σ^2 for the linear model $\mathbf{y} = \mathbf{1}\boldsymbol{\beta} + \varepsilon$, with $\boldsymbol{\beta} = \mu = \mathrm{E}(y_i)$ and $\operatorname{Var}\varepsilon = \sigma^2 I$. (This is sometimes called the *null* model.)

3. An online survey collects data on factors that affect a person's pay rate (per hour). The table below shows pay rate (`pay`) and number of years of education (`yrEdu`) of five participants.

id	pay	yrEdu
1	7.06	9
2	18.93	12
3	20.17	12
4	29.58	16
5	33.90	20

(a) Let x_i and y_i denote the years of education and pay rate of individual i. We want to fit the model $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$. Given that $\sum_i x_i^2 = 1025$, $\bar{x} = 13.8$, $\bar{y} = 21.928$, $\sum_i x_i y_i = 1684.02$, find the least squares estimates of β_0, β_1

Solution This is a simple linear regression model, so using the formulas in the lecture notes:

$$\begin{aligned}
\widehat{\beta}_1 &= \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n} (x_i - \overline{x})^2} \\
&= \frac{\sum_{i=1}^{n} x_i y_i - n\overline{x}\overline{y}}{\sum_{i=1}^{n} x_i^2 - n\overline{x}^2} \\
&= \frac{1684.02 - 5 \times 13.8 \times 21.928}{1025 - 5 \times 13.8^2} \\
&= 2.348736
\end{aligned}$$

$$\widehat{\beta}_0 = \bar{y} - \widehat{\beta}_1 \bar{x} = -10.48456$$

(b) Suppose we have calculate the least squares estimators $\widehat{\beta}_0$ and $\widehat{\beta}_1$ in R. Consider the following R commands and output

```
> pay = c(7.06,18.93,20.17,29.58,33.9)
> yrEdu = c(9,12,12,16,20)
> e = pay - (betahat0 + betahat1*yrEdu)
> t(e)%*%e
[,1]
[1,] 33.41216
```

Calculate the sample variance s^2.

Solution $s^2 = \mathrm{SS}_{\mathrm{Res}}/(n-p) = 33.41216/(5-2) = 11.13739$

(c) Estimate the pay rate of a person with 14 years of education.

Solution $\mathbf{t} = (1, 14)^T$. Then estimated pay rate is $\mathbf{t}^T \widehat{\boldsymbol{\beta}} = 22.39775$.

(d) The leverage of the data points are given as

```
> model1_leverage
[1] 0.5164835 0.2445055 0.2445055 0.2664835 0.7280220
```

Calculate the standardised residual for the 3^{rd} observation.

Solution $e_3 = 20.17 - \widehat{\beta}_0 - \widehat{\beta}_1 \times 12 = 2.46973$.
$z_3 = e_3/\sqrt{s^2(1 - H_{33})} = 2.46973/\sqrt{11.13739 \times (1 - 0.2445055)} = 0.8514166$.

Practical questions

Attempt the exercises below.

1. Last week you wrote a program to calculate $h(x, n)$, the sum of a finite geometric series. Turn this program into a *function* that takes two arguments, x and n, and returns $h(x, n)$.

   Make sure you deal with the case $x = 1$.

   Solution:

```
series_sum <- function(x, n) {
# sum of x^k for k = 0, ..., n
if (x == 1) {
return(n + 1)
} else {
return((x^(n+1) - 1)/(x - 1))
}
}
```

2. Consider the following program

```
# clear the workspace
rm(list=ls())

random.sum <- function(n) {
# sum of n random numbers
x[1:n] <- ceiling(10*runif(n))
cat("x:", x[1:n], "\n")
return(sum(x))
}
```

   Below are the output of the function for $n = 10$ and $n = 5$

```
> x <- rep(100, 10)
> show(random.sum(10))
x: 6 10 7 5 8 6 5 10 9 4
[1] 70
> show(random.sum(5))
x: 8 9 4 5 10
[1] 536
```

   Explain what is going wrong and how you would fix it.

   Solution

   The problem is is the line

```
x[1:n] <- ceiling(10*runif(n))
```

   One way to fix it is

```
random.sum.fix <- function(n) {
# sum of n random numbers
x <- ceiling(10*runif(n))
cat("x:", x[1:n], "\n")
return(sum(x))
}
```

3. In this question we simulate the rolling of a die. To do this we use the function `runif(1)`, which returns a 'random' number in the range (0,1). To get a random integer in the range $\{1, 2, 3, 4, 5, 6\}$, we use `ceiling(6*runif(1))`, or if you prefer, `sample(1:6,size=1)` will do the same job.

(a) Suppose that you are playing the gambling game of the Chevalier de Méré. That is, you are betting that you get at least one six in four throws of a die. Write a program that simulates one round of this game and prints out whether you win or lose.

Check that your program can produce a different result each time you run it.

Solution:

```
win <- FALSE
for (i in 1:4) {
if (sample(1:6, size = 1) == 6) {
     win <- TRUE
   }
 }
if (win) {
   print("win")
 } else {
   print("lose")
 }

## [1] "win"
```

(b) Turn the program that you wrote in part (a) into a function `sixes`, which returns `TRUE` if you obtain at least one six in n rolls of a fair die, and returns `FALSE` otherwise. That is, the argument is the number of rolls n, and the value returned is `TRUE` if you get at least one six and `FALSE` otherwise.

How would you give n the default value of 4?

Solution:

```
sixes <- function(n = 4) {
   # plays the game of the Chevalier de Mere
   # returns TRUE if at least one six in n rolls
   # returns FALSE otherwise
   win <- FALSE
   for (i in 1:n) {
     if (sample(1:6, size = 1) == 6) {
       return(TRUE)
     }
   }
   return(FALSE)
 }
```

Here is a vectorised version.

```
sixes <- function(n = 4) {
   sum(sample(1:6, size = n, replace = TRUE) == 6) > 0
 }
```

(c) Now write a program that uses your function `sixes` from part (b), to simulate N plays of the game (each time you bet that you get at least one six in n rolls of a fair die). Your program should then determine the proportion of times you win the bet. This proportion is an estimate of the *probability* of getting at least one six in n rolls of a fair die.

Run the program for $n = 4$ and $N = 100$, 1000, and 10000, conducting several runs for each N value. How does the *variability* of your results depend on N?

The probability of getting no 6's in n rolls of a fair die is $(5/6)^n$, so the probability of getting at least one is $1 - (5/6)^n$. Modify your program so that it calculates the theoretical probability as well as the simulation estimate and prints the difference between them. How does the *accuracy* of your results depend on N?

Solution:

```r
p_estimate <- function(N, n = 4) {
   # proportion of wins in N runs of sixes(n)
   total_wins <- 0
   for (i in 1:N) {
     if (sixes(n)) total_wins <- total_wins + 1
   }
   return(total_wins/N)
 }
p_accuracy <- function(N, n = 4) {
   # accuracy of p_estimate
   total_wins <- 0
   for (i in 1:N) {
     if (sixes(n)) total_wins <- total_wins + 1
   }
   return(total_wins/N - 1 + (5/6)^n)
 }
```

(d) In part (c), instead of processing the simulated runs as we go, suppose we first store the results of every game in a file, then later postprocess the results.

Write a program to write the result of all N runs to a textfile `sixes_sim.txt`, with the result of each run on a separate line. For example, the first few lines of the textfile could look like

```
# TRUE
# FALSE
# FALSE
# TRUE
# FALSE
# .
# .
```

Now write another program to read the textfile `sixes_sim.txt` and again determine the proportion of bets won.

This method of saving simulation results to a file is particularly important when each simulation takes a very long time (hours or days), in which case it is good to have a record of your results in case of a system crash.

Solution:

```r
sixes_sim <- function(N, n = 4) {
   # runs sixes(n) N times and saves the results in "sixes_sim.txt"
   cat(file="sixes_sim.txt") # deletes contents of file
   for (i in 1:N) {
     cat(file = "sixes_sim.txt", sixes(n), "\n", append = TRUE)
   }
 }
sixes_sim(100)
results <- scan("sixes_sim.txt", what = TRUE)
mean(results)

## [1] 0.59
```

4. Let $\mathbf{y} = \begin{bmatrix} y_1 & y_2 \end{bmatrix}^T$ be a normal random vector with mean and variance

$$\boldsymbol{\mu} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \quad V = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}.$$

Let

$$A = \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

.

From Theorem 3.9 we know that $\mathbf{y}^T A\mathbf{y}$ follows a χ^2 distribution with degree of freedom 1 and noncentrality parameter $\lambda = 4.5$.

Solution: We first need to verify whether AV is idempotent

```
> mu = c(2,4);
> V = 2*diag(2)
> A = 1/4*matrix(c(1,1,1,1),2,2);
> A
     [,1] [,2]
[1,] 0.25 0.25
[2,] 0.25 0.25
>
> (AV = A%*%V)
     [,1] [,2]
[1,]  0.5  0.5
[2,]  0.5  0.5
> AV%*%AV
     [,1] [,2]
[1,]  0.5  0.5
[2,]  0.5  0.5
>
> 1/2*mu%*%A%*%mu
     [,1]
[1,]  4.5
```

(a) Generate $n = 1000$ samples $\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n)}\}$ from $MVN(\boldsymbol{\mu}, V)$.

```
> n = 1000
> ysample = mvrnorm(n,mu,V)
> dim(ysample)
[1] 1000    2
```

(b) Compute $\mathbf{y}^T A\mathbf{y}$ for all $\mathbf{y}^{(i)}$ that we generated.

```
> quadform = function(y, A) t(y) %*% A %*% y
> quadsampleA = apply(ysample,1,quadform, A= A)
> str(quadsampleA)
 num [1:1000] 5.4 15.87 4.77 8.24 12.51 ...
```

(c) Plot the histogram of the $\mathbf{y}^T A\mathbf{y}$ values that we have computed.

```
> hist(quadsampleA, col = 'coral1')
```

(d) Now generate n samples from $\chi^2_{1,4.5}$ distribution using `rchisq()`.

```
> chiA = rchisq(n,1,mu%*%A%*%mu)
```

(e) Plot the histogram of the generated samples on the same graph with the histogram in part (c).
The two histograms should overlap.

```
> hist(chiA,add = T, col = 'lightcyan')
> # not very nice so improve it a little bit by making the color transparent
> # first get the red, green and blue values needed for the rgb() command
> col2rgb(c("coral1",'lightcyan'))
      [,1] [,2]
red    255  224
green  114  255
blue    86  255
> hist(quadsampleA, col = rgb(255,114,86,max = 255,alpha = 125))
> # because rgb color are defined in range 0-255
> hist(chiA, col = rgb(225,255,255,max = 255,alpha = 125),add = T)
```