GU4206-GR5206 Final Exam [100 pts]

Student (UNI)

May 4th, 2018

The STAT Spring 2018 GU4206-GR5206 Final Exam is open notes, open book(s), open computer and online resources are allowed. Students are **not** allowed to communicate with any other people regarding the final with the exception of the instructor (Gabriel Young) and TA (Fan Gao). This includes emailing fellow students, using WeChat and other similar forms of communication. If there is any suspicious of students cheating, further investigation will take place. If students do not follow the guidelines, they will receive a zero on the exam and potentially face more severe consequences. The exam will be posted on Canvas at 9:00AM. Students are required to submit both the .pdf and .Rmd files on Canvas (or .html if you must) by 12:00PM. Late exams will not be accepted. If for some reason you are unable to upload the completed exam on Canvas by 12:00PM, them immediately email the markdown file to the course TA.

# Part 1: Simulation [40 pts]

In this section, we consider a **mixture** of two normal distributions. Here we assume that our random variable is governed by the probability density , defined by where and the parameter space is defined by , , and . The **mixture parameter** governs how much mass gets placed on the first distribution and the complement of governs how much mass gets placed on the other distribution .

In our setting, suppose that we are simulating heights from the population of both males and females. Assume that males are distributed normal with mean and standard deviation and females are distributed normal with mean and standard deviation . Also assume that each distribution contributes equal mass, i.e., set the mixture parameter to . The distribution of males is governed by

and the distribution of females is governed by

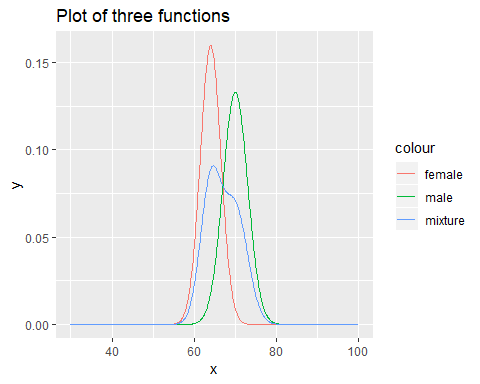
The goal is to **simulate** from the **mixture distribution**

where using the accept-reject algorithm.

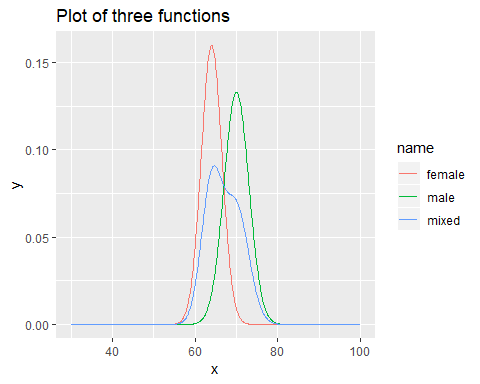
## Perform the following tasks:

1. [10 pts] Using **ggplot**, graph , and the mixture all on the same plot. Make sure the plot includes a legend and is labeled appropriately.

# Solution goes here   
  
library(ggplot2)  
x <- seq(30,100,by=0.1)  
f1 <- dnorm(x,mean=70,sd=3)  
  
f2 <- dnorm(x,mean=64,sd=2.5)  
  
# 目标函数f要写成函数 因为后续会用到  
f <- function(x){  
 0.5\*dnorm(x,mean=70,sd=3)+ 0.5\*dnorm(x,mean=64,sd=2.5)  
}  
  
  
  
# ggplot绘图的简便方法：  
# 1、数据准备：把x和所有y直接写入一个dataframe中，再把列名写为图例中想要的列名  
# 2、绘图：分开来画，每一个col都放在aes()内，且一列名命名  
new <- data.frame(x,f1,f2,f(x))  
colnames(new) = c("x","male","female","mixture")  
ggplot(data = new) +   
 geom\_line(mapping = aes(x=x, y = male, col = "male")) +   
 geom\_line(mapping = aes(x=x, y = female,col = "female")) +   
 geom\_line(mapping = aes(x=x, y = mixture,col = "mixture"))+  
 labs(title = "Plot of three functions", x = "x", y = "y")



#ggolot画图准备工作：先把数据都写到dataframe中，x和name需要以重复形式给出--------  
data <- data.frame(x=c(x,x,x),  
 y=c(f1,f2,f(x)),  
 name = c(rep("male",length(x)),rep("female",length(x)),rep("mixed",length(x))))  
# 图例的名字即为这里第三列的列名  
  
  
library(ggplot2)  
ggplot(data = data) +  
 geom\_line(mapping = aes(x = x, y = y, col= name)) +   
 labs(title = "Plot of three functions", x = "x", y = "y")



1. [25 pts] Use the **accept-reject** algorithm to simulate from this mixture distribution. To receive full credit:

2.i) [5 pts]

# Solution goes here   
g <- function(x){  
 result <- dnorm(x,mean=65,sd=5) #通过后面的绘图，来调整mean和sd  
 return(result)  
}  
  
# 找最大值的方法：  
x[which.max(f(x))]

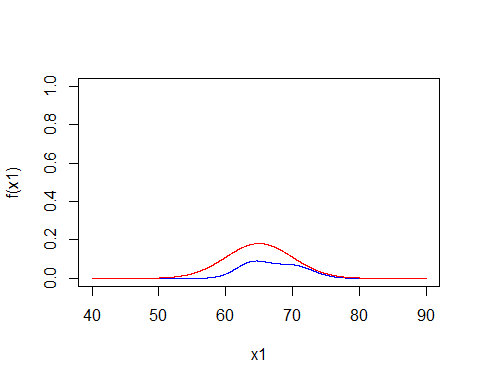
## [1] 64.7

f(x[which.max(f(x))])

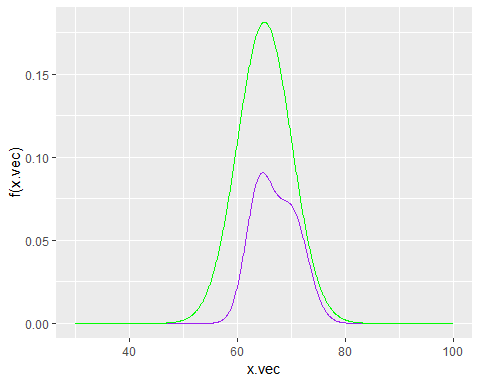
## [1] 0.09068553

2.ii) [10 pts] By inspection, we see that choosing allows to be greater than for all and the envelope sits relatively close to the target distribution.

# Solution goes here   
  
# alpha要写死 或者给出默认值  
e <- function(x,alpha=0.44){  
 ex <- g(x)/alpha  
 return(ex)  
}  
  
x1 <- seq(40,90,0.01)  
  
plot(x1,f(x1),col="blue",type="l",ylim = c(0,1))  
lines(x1,e(x1,0.44),col="red")



#-----------用ggplot2 画图  
x.vec <-seq(30,100,by=.1)  
ggplot()+  
 geom\_line(mapping =aes(x = x.vec, y =f(x.vec)),col="purple")+  
 geom\_line(mapping =aes(x = x.vec, y =e(x.vec)),col="green")



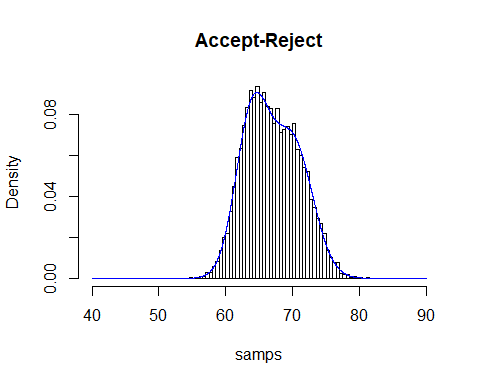
2.iii) [10 pts] The **Accept-Reject** algorithm is coded below:

# Solution goes here   
  
n.samps <- 10000 # Samples desired  
n <- 0 # counter for number samples accepted  
samps <- numeric(n.samps) # initialize the vector of output  
while (n < n.samps) {  
 y <- rnorm(1,65,5) # random draw from g  
 u <- runif(1)   
 if (u < f(y)/e(y)) {  
 n <- n + 1  
 samps[n] <- y  
 }  
}  
  
head(samps)

## [1] 68.56337 64.57122 65.08973 69.71352 67.59701 60.27149

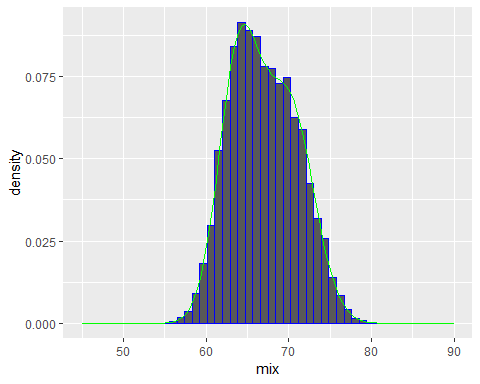
Plot:

# Solution goes here   
  
hist(samps,breaks=50,probability = TRUE,main="Accept-Reject",xlim=c(40,90))  
lines(x1,f(x1),col="blue")



#---------------ggolot2--------------  
mix.sim <-data.frame(mix=samps)  
ggplot(mix.sim)+  
 geom\_histogram(mapping=aes(x=mix,y=..density..),col="blue",bins=50)+  
 xlim(45,90)+ #对x加上限制条件  
 stat\_function(fun=f,colour="green")

## Warning: Removed 2 rows containing missing values (geom\_bar).



1. [5 pts] Slightly change the **Accept-Reject** algorithm from Part (2.iii) to also include the acceptance rate, i.e., how many times did the algorithm accept a draw compared to the total number of trials performed. What proportion of cases were accepted? Compare this number to your chosen and comment on the result.

# Solution goes here   
  
  
f <- function(x){  
 0.5\*dnorm(x,mean=70,sd=3)+ 0.5\*dnorm(x,mean=64,sd=2.5)  
}  
  
# Accept-Reject中，n为接受的数量（因为他已经是写在if后面了）  
  
  
n.samps <- 10000 # Samples desired  
n <- 0 # counter for number samples accepted  
total\_n <- 0  
samps <- numeric(n.samps) # initialize the vector of output  
while (n < n.samps) {  
 total\_n <- total\_n + 1  
 y <- rnorm(1,65,5) # random draw from g  
 u <- runif(1)   
 if (u < f(y)/e(y)) {  
 n <- n + 1  
 samps[n] <- y  
 }  
}  
  
acceptance\_rate <- n.samps/total\_n  
acceptance\_rate

## [1] 0.4440497

# it is really close to alpha

# Part II: Maximum Likelihood Estimaiton and Newton’s Method [25 pts]

Recall in logistic regression, the likelihood function is derived by **linking** the mean of with a linear function.

**Logistic Regression Model:**

Let be independently distributed Bernoulli random variables with respective success probabilities . Then the **logistic regression model** is:

Notice with some simple algebra, the above model can be expresses as:

$$
\log \bigg{(} \frac{p\_i}{1-p\_i}\bigg{)}=\beta\_0+\beta\_1x\_{i1}+\beta\_2x\_{i2}+\cdots+\beta\_px\_{i,p} x\_i, \ \ \ \, i=1,2,\ldots,n.
$$

The main idea is to **link** the expected value of () to a linear function. This same principle can be applied to other settings.

## Data Description

Consider a geriatrics experiment designed as a prospective study to investigate the effects of two interventions on the frequency of falls. One hundred subjects were randomly assigned to one of the two interventions: education only () and education plus aerobic exercise training (). Subjects were at least 65 years of age and in reasonably good health. Three variables considered to be important as control variables were gender (female, 1=male), a balance index (), and a strength index (). The higher the balance index, the more stable the subject: and the higher the strength index, the stronger the subject. Let be the number of falls during the six month study.

setwd("C:/Users/Alan\_/Desktop/Semester1 Courses/STAT computing/sample final")  
glm.data <- read.table("CH14PR39.txt")  
names(glm.data) <- c("Y","X1","X2","X3","X4")  
head(glm.data)

## Y X1 X2 X3 X4  
## 1 1 1 0 45 70  
## 2 1 1 0 62 66  
## 3 2 1 1 43 64  
## 4 0 1 1 76 48  
## 5 2 1 0 51 72  
## 6 1 1 1 73 39

In this setting, the response variable takes on discrete count values, therefore it is reasonable to assume are independent Poisson random variables with mean . Here we can choose a link function that relates to the linear function .

**Perform the follwoing task:**

1. [25 pts] Assume are independent Poisson random variables with mean
2. Note that the link function is . Use maximum likelihood estimation to estimate the Poisson regression model. To receive full credit:

4.i) [10 pts]

# Solution goes here  
# 求似然函数：doo(Y的数据、参数、log=TRUE)  
  
pois.neg.ll <- function(beta){  
 lambda <- exp(beta[1]+  
 glm.data$X1\*beta[2]+  
 glm.data$X2\*beta[3]+  
 glm.data$X3\*beta[4]+  
 glm.data$X4\*beta[5])  
 neg.ll <- -sum(dpois(glm.data$Y, lambda, log = TRUE))  
 return(neg.ll)  
}

4.ii) [5 pts]

# Solution goes here   
pois.neg.ll(beta=rep(0,5))

## [1] 362.8506

4.iii) [10 pts]

# Solution goes here   
library(numDeriv)  
Newtons.Method <- function(f, x0, max.iter = 200, stopping.deriv = 0.01, ...) {  
   
 n <- length(x0)  
 xmat <- matrix(0, nrow = n, ncol = max.iter)  
 xmat[,1] <- x0  
 for (k in 2:max.iter) {  
 # Calculate the gradient  
 grad.cur <- grad(f, xmat[ ,k-1], ...)  
 # the hessian use hessian to be the step\_length of each step  
 hess.cur <- hessian(f, xmat[, k-1], ...)  
 # Should we stop?  
 if (all(abs(grad.cur) < stopping.deriv)) {  
 k <- k-1  
 break  
 }  
 # Move in the opposite direction of the grad solve():get inverse of matrix  
 xmat[ ,k] <- xmat[ ,k-1] - solve(hess.cur) %\*% grad.cur  
 }  
   
 xmat <- xmat[ ,1:k] # Trim  
 return(list(x = xmat[,k], xmat = xmat, k = k, minimum=f(xmat[,k],...)))  
}  
  
out <- Newtons.Method(pois.neg.ll, x0 = rep(0,5))   
out$x

## [1] 0.489467146 -1.069402560 -0.046606074 0.009469987 0.008565829

out$k

## [1] 8

#----------------------------------------------------------------------

# 如果数据集是写死的，nlm(函数、初始参数)即可  
nlm(pois.neg.ll,p=rep(0,5))

## Warning in nlm(pois.neg.ll, p = rep(0, 5)): NA/Inf replaced by maximum  
## positive value  
  
## Warning in nlm(pois.neg.ll, p = rep(0, 5)): NA/Inf replaced by maximum  
## positive value  
  
## Warning in nlm(pois.neg.ll, p = rep(0, 5)): NA/Inf replaced by maximum  
## positive value

## $minimum  
## [1] 183.6439  
##   
## $estimate  
## [1] 0.489698705 -1.069396059 -0.046623915 0.009468908 0.008563197  
##   
## $gradient  
## [1] 3.272476e-06 -1.179104e-06 3.683454e-06 2.938202e-04 1.957336e-04  
##   
## $code  
## [1] 2  
##   
## $iterations  
## [1] 56

# 如果数据集没有写死，则后面还要加上数据参量（数据为整个数据集）  
# nlm(neg.gamma.ll, c(19, 1), data = cats$Hwt)

Check the result with **glm()** (optional)

glm(Y~X1+X2+X3+X4,data=glm.data,family="poisson")

##   
## Call: glm(formula = Y ~ X1 + X2 + X3 + X4, family = "poisson", data = glm.data)  
##   
## Coefficients:  
## (Intercept) X1 X2 X3 X4   
## 0.489467 -1.069403 -0.046606 0.009470 0.008566   
##   
## Degrees of Freedom: 99 Total (i.e. Null); 95 Residual  
## Null Deviance: 199.2   
## Residual Deviance: 108.8 AIC: 377.3

##### These two results are almost the same.

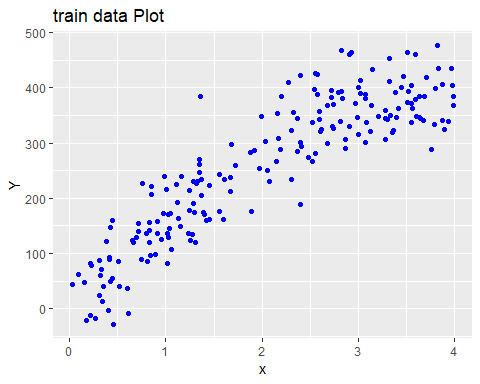
# Part III: Cross Validadtion and Linear Regression [35 pts]

The goal of this section is to illustrate how the **degree** of a polynomial model can be thought of as a tuning parameter.

**Perform the follwoing task:**

1. [5 pts] Upload the dataset **finalexamtrain.csv** and plot versus using **ggplot**. Change the points to blue in the plot.

train <- read.csv("finalexamtrain.csv")  
  
ggplot(data = train) +  
 geom\_point(mapping = aes(x = x, y = Y),col="blue") +  
 labs(title = "train data Plot", x = "x", y = "Y")



1. [5 pts] In this setting, we regress the response variable against a single predictor using five different models:

**Linear Model: degree=1**

**Quadratic Model: degree=2**

**Cubic Model: degree=3**

**Quartic Model: degree=4**

**Quintic Model: degree=5**

Fit each model using the training data. To make your life easier, I recommend using the Inhibit Interpretation function **I()**. Display the estimated coefficients for each model.

# Solution goes here   
Linear <- lm(Y~x, data = train)  
Quadratic <- lm(Y~x+I(x^2), data = train)  
Cubic <- lm(Y~x+I(x^2)+I(x^3), data = train)  
Quartic <- lm(Y~x+I(x^2)+I(x^3)+I(x^4), data = train)  
Quintic <- lm(Y~x+I(x^2)+I(x^3)+I(x^4)+I(x^5), data = train)  
  
  
## model.quad$coefficients

1. [10 pts] Consider the dataframe **finalexamtest.csv** that contains four validation (test) sets each consisting of 50 cases. Notice that the variable **ValSet** contains four levels. More specifically, the first 50 cases belong to **TestSet1**, the next 50 cases belong to **TestSet2**, …etc.

data.test <- read.csv("finalexamtest.csv")  
head(data.test)

## x Y ValSet  
## 1 0.67 45.00438 TestSet1  
## 2 3.23 386.60818 TestSet1  
## 3 1.54 284.04469 TestSet1  
## 4 2.42 391.53059 TestSet1  
## 5 0.50 1.37337 TestSet1  
## 6 1.18 262.60680 TestSet1

tail(data.test)

## x Y ValSet  
## 195 1.00 147.2985 TestSet4  
## 196 3.12 296.8062 TestSet4  
## 197 3.42 302.9193 TestSet4  
## 198 2.09 263.6752 TestSet4  
## 199 1.69 212.8365 TestSet4  
## 200 1.65 303.2247 TestSet4

levels(data.test$ValSet)

## [1] "TestSet1" "TestSet2" "TestSet3" "TestSet4"

Write a function named **test.error** that inputs a dataframe and outputs the **test error** for each model. The function should give four predictions errors corresponding to each model, i.e., linear, quadratic, cubic, quartic, and quintic. Note that you are still using the trained models from Part (6). You are allowed to hard code some of this function. Test the \*\*test.error\* function on the full training data.

**Hint: y.test <- predict(model,newdata = data.test[“x”])**

# Solution goes here   
test.error <- function(data=train){  
 Linear.test <- predict(Linear,newdata = data["x"])  
 Quadratic.test <- predict(Quadratic,newdata = data["x"])  
 Cubic.test <- predict(Cubic,newdata = data["x"])  
 Quartic.test <- predict(Quartic,newdata = data["x"])  
 Quintic.test <- predict(Quintic,newdata = data["x"])  
   
 Linear.error <- mean((Linear.test-data$Y)^2)  
 Quadratic.error <- mean((Quadratic.test-data$Y)^2)  
 Cubic.error <- mean((Cubic.test-data$Y)^2)  
 Quartic.error <- mean((Quartic.test-data$Y)^2)  
 Quintic.error <- mean((Quintic.test-data$Y)^2)  
   
 return(c(Linear.error,Quadratic.error,Cubic.error,  
 Quartic.error,Quintic.error))  
}  
  
test.error(train)

## [1] 3211.680 2485.083 2452.323 2450.637 2446.708

1. [10 pts] Use the **Split/Apply/Combine** model to compute the test error for each test set: **TestSet1**, **TestSet2**, **TestSet3**, **TestSet4**. You can exhaustively perform this task for partial credit. Display the test errors for each test set and each model. Also display the average test error over each validation set.

# Solution goes here   
# 只是说Split/Apply/Combine，没有说其他的，就还是用第二种好啦  
# apply()函数中，1是对列求值，2是对行求值  
  
  
library(plyr)  
five\_error <- daply(data.test, .(ValSet), test.error)  
five\_error

##   
## ValSet 1 2 3 4 5  
## TestSet1 3611.566 3324.065 3068.520 3074.860 3091.755  
## TestSet2 3466.068 2832.745 2797.414 2804.055 2828.324  
## TestSet3 3434.612 3087.183 2951.258 2973.750 3068.169  
## TestSet4 2524.945 1687.839 1794.613 1795.619 1807.527

apply(five\_error,2,mean)

## 1 2 3 4 5   
## 3259.298 2732.958 2652.951 2662.071 2698.944

##--------------------------------------  
  
data.split <- split(data.test, data.test$ValSet)  
names(data.split)

## [1] "TestSet1" "TestSet2" "TestSet3" "TestSet4"

five\_error2 <- sapply(data.split,test.error)  
five\_error2

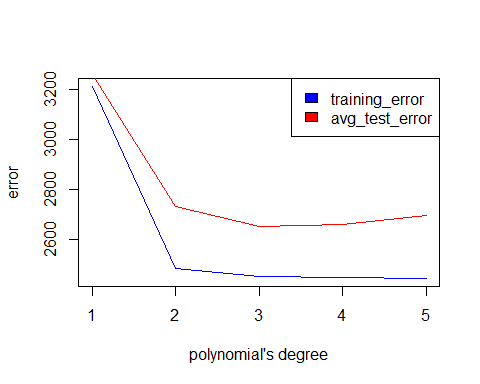
## TestSet1 TestSet2 TestSet3 TestSet4  
## [1,] 3611.566 3466.068 3434.612 2524.945  
## [2,] 3324.065 2832.745 3087.183 1687.839  
## [3,] 3068.520 2797.414 2951.258 1794.613  
## [4,] 3074.860 2804.055 2973.750 1795.619  
## [5,] 3091.755 2828.324 3068.169 1807.527

apply(five\_error2, 1, mean)

## [1] 3259.298 2732.958 2652.951 2662.071 2698.944

1. [5 pts] Create a plot (base R or ggplot) showing both the **average test error** and the **training error** as a function of the polynomial’s **degree**. Briefly comment on this plot.

# Solution goes here   
training\_error <- test.error(train)  
avg\_test\_error <- apply(five\_error,2,mean)  
  
plot(c(1:5),training\_error,col="blue",type="l",xlab = "polynomial's degree",ylab = "error")  
lines(c(1:5),avg\_test\_error,col="red")  
legend("topright", legend = c("training\_error","avg\_test\_error"),  
 fill = c("blue","red"))



##### The linear degree=1 model does a poor job on both the training and test datasets. The test error achievesits minimum at degree 2. The training error continues to decrease as the degree increases. As the degreeincreases, we expect for the test error to increase and the training error to decrease.