Lab2 Computational Statistics

Andreas Christopoulos Charitos (andch552) Omkar Bhutra (omkbh878)

27 jan 2019

Question 1

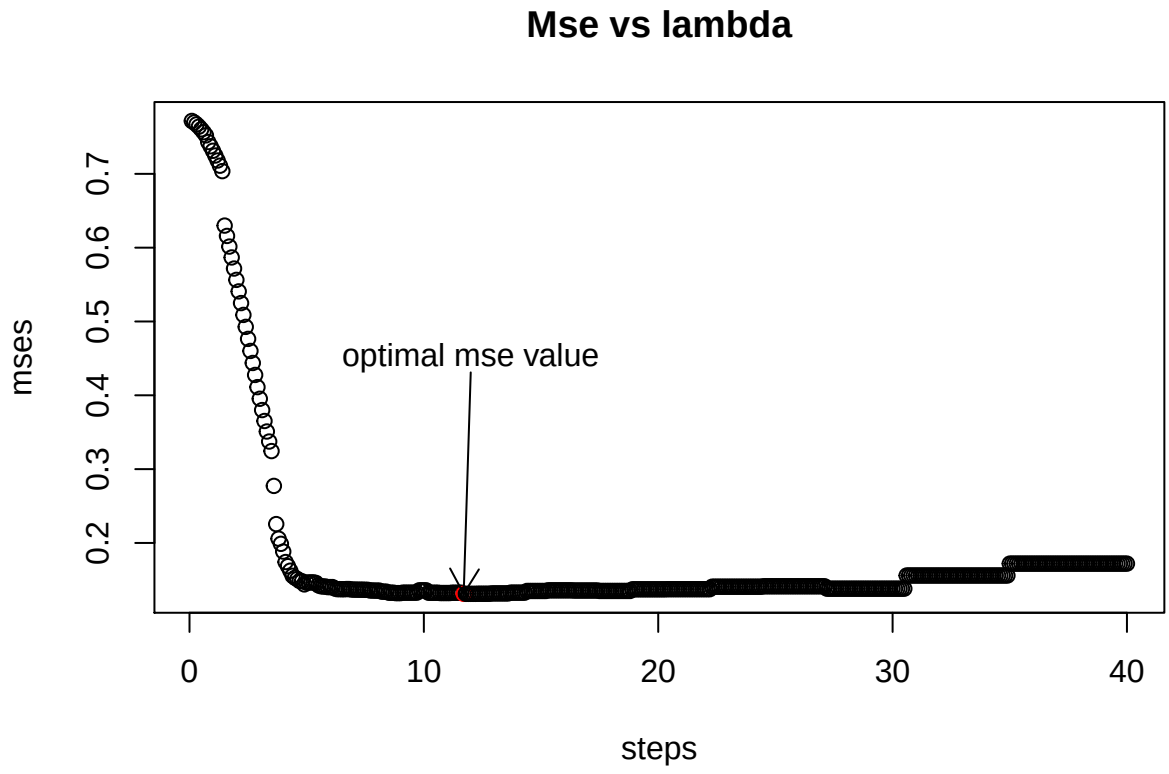
1

```
#importing and diving data
mortality<-read.csv2("mortality_rate.csv")
mortality$LMR<-log(mortality$Rate)

n=dim(mortality)[1]
set.seed(123456)
id=sample(1:n , floor(n*0.5))
train=mortality[id, ]
test=mortality[-id, ]
```

```
#defining the function "myMSE"

myMSE<-function(lambda,pars,iterCounter = T){
  model<-loess(pars$Y~pars$X,enp.target = lambda)
  preds<-predict(model,newdata=pars$X_test)
  mse<-sum((pars$Y_test-preds)^2)/length(pars$Y_test)

  # If we want a iteration counter
  if(iterCounter){
    if(!exists("iterForMyMSE")){
      # Control if the variable exists in the global environemnt,
      # if not, create a variable and set the value to 1. This
      # would be the case for the first iteration
      # We will call the variable 'iterForMyMSE'
      assign("iterForMyMSE",
             value = 1,
             globalenv())
    } else {
      # This part is for the 2nd and the subsequent iterations.
      # Starting of with obtaining the current iteration number
      # and then overwrite the current value by the incremental
      # increase of the current value
      currentNr <- get("iterForMyMSE")
      assign("iterForMyMSE",
             value = currentNr + 1,
             globalenv())
    }
  }
  return(mse)
}
```

```
set.seed(123456)
steps=seq(0.1,40,0.1)
```

```
mses<-double(length(steps))

mypars<-list(X=train$Day,Y=train$LMR,X_test=test$Day,Y_test=test$LMR)

for(i in steps){
  mses[which(i==steps)]<-myMSE(i,mypars)

}

optimal_l=steps[which.min(mses)]
optimal_mse=min(mses)

plot(x=steps,y= mses,main = "Mse vs lambda",
      col=ifelse(steps==optimal_l,"red","black"))
arrows(optimal_l+0.3, optimal_mse+0.3, x1 = optimal_l, y1 = optimal_mse,
               length = 0.15, angle = 30)
text(optimal_l+0.3,optimal_mse+0.32,labels=c("optimal mse value"))
```




```
iters1=iterForMyMSE

results1=list("par"=optimal_l,"value"=optimal_mse)

cat("The number of iterations using brute force are :", iters1,"\n",
    "The results from the brute force are :\n")
```

```
## The number of iterations using brute force are : 400
```

```
##  The results from the brute force are :
```

```
results1
```

```
## $par
## [1] 11.7
##
## $value
## [1] 0.131047
```

5

```
set.seed(123456)
remove("iterForMyMSE")
xmin <- optimize(myMSE,pars=mypars,
                 interval=c(0.1,40),maximum  = FALSE,tol=0.01)
iters2=iterForMyMSE
cat("The number of iterations using lambda in [0.1,40] and accuracy 0:01 are :", iters2,"\n",
    "The output of optimize function is : \n")
```

```
## The number of iterations using lambda in [0.1,40] and accuracy 0:01 are : 18
##  The output of optimize function is :
```

```
## $minimum
## [1] 10.69361
##
## $objective
## [1] 0.1321441
```

We can see comparing the results with the previous step that the number of iterations needed for the algorithm to converge decreased dramatically from 400 to only 18.

6

```
set.seed(123456)
remove("iterForMyMSE")
xmin1=optim(c(35), myMSE,pars=mypars,
      method = c( "BFGS"))

iters3=iterForMyMSE
cat("The new number of iterations using BFGS algorithm and lambda = 35 are :", iters3,"\n",
    "The output of optimize function is : \n")
```

```
## The new number of iterations using BFGS algorithm and lambda = 35 are : 3
##  The output of optimize function is :
```

```
## $par
## [1] 35
##
## $value
## [1] 0.1719996
##
## $counts
## function gradient
##        1        1
##
```

```
## $convergence
## [1] 0
##
## $message
## NULL
```

Comparing again the number of iterations with the previous step we can see that the iterations decreased from 18 to only 3 using BFGS.

The table below summarises the results of the 3 methods used

```
library(knitr)

sumtable<-data.frame("Brute Force Method"=c(optimal_l,optimal_mse,iters1),
                     "Optimize function"=c(xmin$minimum,xmin$objective,iters2),
                     "Optim function BFGS"=c(xmin1$par,xmin1$value,iters3))
rownames(sumtable)<-c("lambda","mse","iters")

kable(sumtable)
```

	Brute.Force.Method	Optimize.function	Optim.function.BFGS
lambda	11.700000	10.6936107	35.0000000
mse	0.131047	0.1321441	0.1719996
iters	400.000000	18.0000000	3.0000000

As we can see from the table above the BFGS algorithm is able to converge with only 3 iterations which are lower compared to the other 2 methods but with a little higher mse value.

Question 2

2

Maximum Likehihood estimation

The probability function is given by the formula

$$f(x) = \frac{1}{\sqrt{(2\pi\sigma^2)}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

Assuming that the observations from the sample are i.i.d. the likelihood formula is given by

$$f(x_1, x_2, ..., x_n|\mu, \sigma^2) = \prod_{j=1}^{n} (2\pi\sigma^2)^{-(n/2)} exp(-\frac{(x_j-\mu)^2}{2\sigma^2})$$

which is equavalent to :

$$f(x_1, x_2, ..., x_n|\mu, \sigma^2) = (2\pi\sigma^2)^{-(n/2)} exp(-\frac{\sum_{j=1}^{n}(x_j-\mu)^2)}{2\sigma^2})$$

The loglikekihood function is then

$$L = \ln[(2\pi\sigma^2)^{(-n/2)} exp(-(\frac{1}{2\sigma^2})\sum_{j=1}^{n}(x_j-\mu)^2)] =$$

$$\ln((2\pi\sigma^2)^{(-n/2)}) + \ln(exp(-(\frac{1}{2\sigma^2})\sum_{j=1}^{n}(x_j-\mu)^2)) =$$

$$-\frac{n}{2}\ln(2\pi\sigma^2) - (\frac{1}{2\sigma^2})\sum_{j=1}^{n}(x_j-\mu)^2 =$$

$$-\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - (\frac{1}{2\sigma^2})\sum_{j=1}^{n}(x_j-\mu)^2$$

Getting the derivative with respect to μ to we get :

$$\frac{\partial L}{\partial \mu} => -\frac{1}{2\sigma^2}(\sum_{j=1}^{n}(x_j^2 - 2x_j\mu + \mu^2))' =>$$

$$-\frac{1}{2\sigma^2}(\sum_{j=1}^{n}x_j^2 - 2\sum_{j=1}^{n}x_j\mu + n\mu^2)' =>$$

$$-\frac{1}{2\sigma^2}(-2\sum_{j=1}^{n}x_j + 2n\mu) => -\frac{1}{2\sigma^2}(\sum_{j=1}^{n}x_j - n\mu)(-2)(e.1)$$

Setting $(e.1)$ equal to zero we get

$$\frac{\partial L}{\partial \mu} = 0 => \mu = \frac{\sum_{j=1}^{n}x_j}{n}$$

Which is the estimator for the parameter $\hat{\mu}$

Getting the derivative with respect to σ we get :

$$\frac{\partial L}{\partial \sigma} => -\frac{n}{2}\frac{1}{\sigma^2}2\sigma - \frac{1}{2}\sum_{j=1}^{n}(x_j-\mu)^2\frac{\theta L}{\theta\sigma}(\sigma^2)^{-1} => -\frac{n}{\sigma} - \frac{1}{2}\sum_{j=1}^{n}(x_j-\mu)^2\frac{-2}{(\sigma^3)} => -\frac{n}{\sigma} + \sum_{j=1}^{n}(x_j-\mu)^2\frac{1}{(\sigma^3)}(e.2)$$

Setting $(e.2)$ equal to zero we get

$$\frac{\partial L}{\partial \sigma} = 0 => \sum_{j=1}^{n}(x_j-\mu)^2 = \frac{n\sigma^3}{\sigma} => \sigma^2 = \frac{\sum_{j=1}^{n}(x_j-\mu)^2}{n} => \sigma = \sqrt{\frac{\sum_{j=1}^{n}(x_j-\mu)^2}{n}}$$

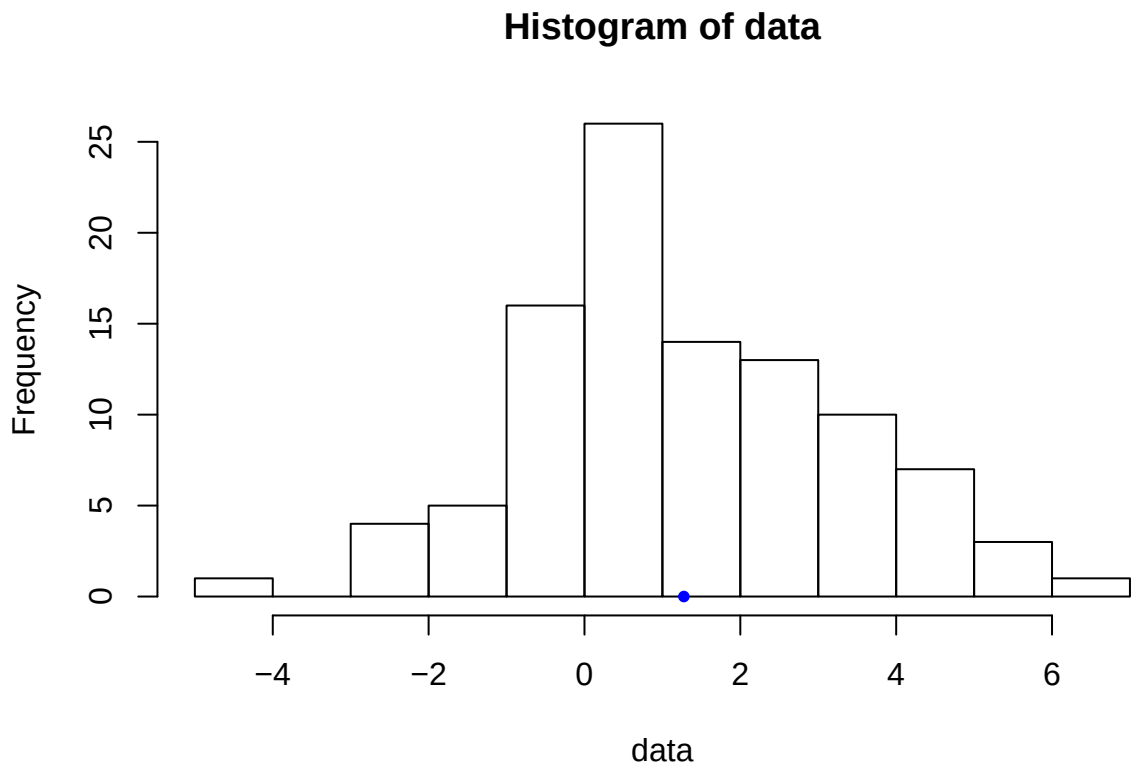
We can use the obtained formulas to obtain the mean $\hat{\mu}$ and variance $\hat{\sigma}$ analytically.

```
#load data
load("data.RData")

#make estimators function
estimators<-function(data){
  n<-length(data)
  mean<-sum(data)/n
  sigma<-sum((data-mean)^2)/n
  return(c("mean"=mean,"sd"=sqrt(sigma)))
```

```
  }


res=estimators(data)

cat("============================================================\n",
    "The obtained mean is :",res["mean"],"and the sd is :",res["sd"])
```

```
## ============================================================
##  The obtained mean is : 1.275528 and the sd is : 2.005976
```

```
hist(data)
points(res["mean"],0,col="blue",pch=20,
       main="Histogram of data")
```



The above histogram shows the distribution of our data and the blue point is indicating the mean value.

3

First we optimize minus loglikehood without specifing gradient

```
loglike<-function(args,x){

  n<-length(x)
  logminus<- (-n*log(2*pi*args[2]^2)/2)-(sum((args[1]-x)^2)/(2*args[2]^2))
  logminus<- -1*logminus
  return(logminus)
```

```
}


#with BFGS
myres1<-optim(c(0,1),fn=loglike,x=data,method = "BFGS")

#with Conjugate Gradient
myres2<-optim(c(0,1),fn=loglike,x=data,method = "CG")


cat("====================================\n",
    "The output of optim with BFGS is :\n")
```

```
## ====================================
##  The output of optim with BFGS is :
```

```
myres1
```

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       41       15
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
cat("\n")
```

```
cat("================================================\n",
    "The output of optim with Conjugate Gradient is :\n")
```

```
## ================================================
##  The output of optim with Conjugate Gradient is :
```

```
myres2
```

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##      208       35
##
## $convergence
## [1] 0
```

```
##
## $message
## NULL
```

Second we optimize minus loglikehood specifing gradient

The gradient with respect to μ is given by the formula (e.1) we calculated erlier $\frac{\partial L}{\partial \mu} = \frac{\sum_{j=1}^{n} x_j - n\mu}{\sigma^2}$ and the derivative with respect to σ is given by the formula (e.2) $\frac{\partial L}{\partial \sigma} = -\frac{n}{\sigma} + \frac{\sum_{j=1}^{n}(x_j-\mu)}{\sigma^3}$

```
gradient<-function(args,x){
  n<-length(x)
  gr_mu<- (sum(x)-n*args[1])/(args[2]^2)

  gr_sigma<- sum((x-args[1])^2)/args[2]^3-n/args[2]

  return(c(-gr_mu,-gr_sigma))
}

#with BFGS
myres3<-optim(c(0,1),fn=loglike,gr=gradient,x=data,method = "BFGS")

#with Conjugate Gradient
myres4<-optim(c(0,1),fn=loglike,gr=gradient,x=data,method = "CG")


cat("==================================================\n",
    "The output of optim with BFGS with gradient is :\n")
```

```
## ==================================================
##  The output of optim with BFGS with gradient is :
```

```
myres3
```

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       39       15
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
cat("\n")
```

```
cat("==================================================================\n",
    "The output of optim with Conjugate Gradient with gradient is :\n")
```

```
## ==================================================================
##  The output of optim with Conjugate Gradient with gradient is :
```

```
myres4
```

```
## $par
## [1] 1.275528 2.005976
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       53       17
##
## $convergence
## [1] 0
##
## $message
## NULL
```

As we can see loglikelihood is monotonically increasing function and is has the same relations of order as the likelihood $p(x|\theta_1) > p(x|\theta_2) <=> \ln(p(x|\theta_1)) > \ln(p(x|\theta_2))$ so maximizing likelihood is equivalent to maximizing log likelihood.The reason why we use $ln(x)$ is for computational convenience as we see in the calculations of loglikehood we did before.Using $ln()$ we where able to discard the exponent and also use the $ln(ab) = \ln(a) + \ln(b)$ we manage to replace multiplication with summation which is more coninient and we are able to maximize just by setting the derivatives to 0.

4

```
library(knitr)

sumdat<-data.frame("BFGS_without_gr"=c(myres1$par[1],myres1$par[2],
                                       myres1$counts[1],myres1$counts[2],myres1$convergence),
                   "CG_without_gr"=c(myres2$par[1],myres2$par[2],
                                     myres2$counts[1],myres2$counts[2],myres2$convergence),
                   "BFGS_with_gr"=c(myres3$par[1],myres3$par[2],
                                    myres3$counts[1],myres3$counts[2],myres3$convergence),
                   "CG_with_gr"=c(myres4$par[1],myres4$par[2],
                                  myres4$counts[1],myres4$counts[2],myres4$convergence)
                   )

rownames(sumdat)<-c("mean","sd","iterations function","iterations gradient","convergence")
sumdat[5,]<-ifelse(sumdat[5,]==0,"yes","no")

kable(sumdat)
```

	BFGS_without_gr	CG_without_gr	BFGS_with_gr	CG_with_gr
mean	1.27552755151932	1.27552771909709	1.27552755040258	1.27552759112531
sd	2.00597696486639	2.00597650338868	2.00597654945241	2.00597647249389
iterations function	41	208	39	53
iterations gradient	15	35	15	17
convergence	yes	yes	yes	yes

The table provides summary information using the 2 algorithms ("BFGS","CG") with and without specifing gradient function in optim function. All the algorihms converge as we can see and the results are almost

identical regarding the mean and the sd.The settings that we would recommend are using BFGS algorithm with gradient because as we can see the number of iterations for both function and gradient are lower compared with the other settings.