HW3

2022-03-02

1

1

```
#' estimate parameters for the logisitic regression
#' @param Y a vector with n 0-1 entry
#' @param X data
#' @param mod estimate mod
#' @param alpha alpha for the estimate
#' @return estimated parameters, CI, log Likelihood

op.log=function(Y,X,mod,alpha=1){
  X.nrow=nrow(X)
  X=cbind(matrix(rep(1,X.nrow)),as.matrix(X))
  X.ncol=ncol(X)
  v.beta=matrix(rep(0,X.ncol))
  error=10
  l=c(0)
  p=1/(1+exp(-X%*%v.beta))
  while(error>0.1){
    beta.old=v.beta
    if (mod=='Newton'){
      #W=-p%*%t(1-p)
      W=diag(c(p*(1-p)))
      At=-t(X)%*%W%*%X
      alpha=1
    }
    else if(mod=='Gradient'){
      At=diag(rep(1,X.ncol))
      colnames(At)=colnames(X)
    }
    v.beta=v.beta-alpha*solve(At)%*%t(X)%*%(Y-p)
    p=1/(1+exp(-X%*%v.beta))
    l=append(l,sum(Y*log(min(p,1-1e-100)))+sum((1-Y)*log(max(1-p,1e-100))))
    #l=append(l,sum(Y*log(p))+sum((1-Y)*log(1-p)))
    error=abs(l[length(l)]-l[length(l)-1])
  }
  M=t(X)%*%diag(c(p*(1-p)))%*%X
  result=data.frame(value=v.beta,
                    CI.L=v.beta+qnorm(0.025,0,1)*matrix(sqrt(1/diag(M))),
                    CI.H=v.beta+qnorm(0.975,0,1)*matrix(sqrt(1/diag(M))))
  return(list(result,l[-1]))
}
```

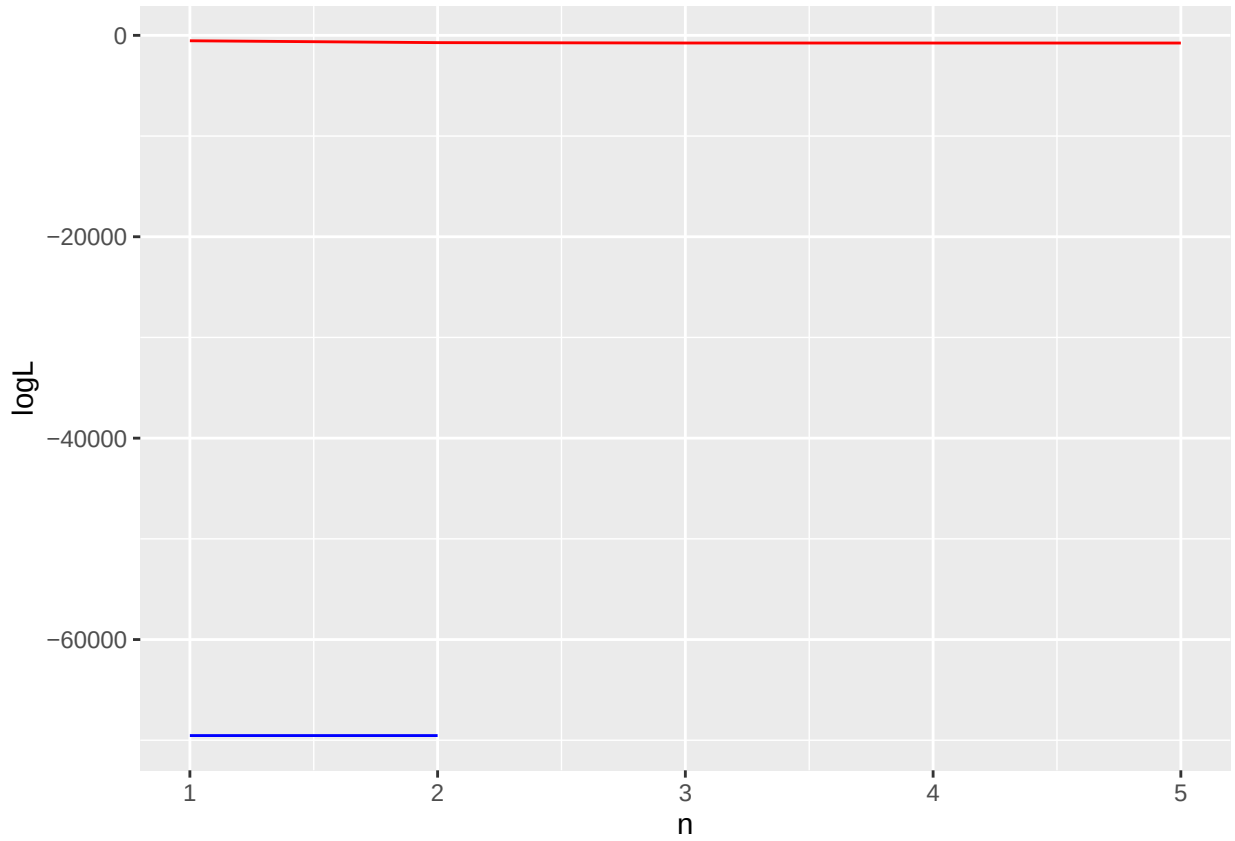
2

```
data=read.csv('https://hastie.su.domains/ElemStatLearn/datasets/SAheart.data',
              head=T,row.names=1)
#data$famhist=as.factor(data$famhist)
data$famhist[data$famhist=='Present']= 1
data$famhist[data$famhist=='Absent']=0
data$famhist=as.numeric(data$famhist)


nt.op=op.log(data$chd,data[,1:9],'Newton')
gr.op=op.log(data$chd,data[,1:9],'Gradient',alpha=.1)
```

3

```
library(ggplot2)
ggplot(data=data.frame(logL=nt.op[[2]],n=1:length(nt.op[[2]])),
       aes(x=n,y=logL))+
  geom_line(color='red')+
  geom_line(data=data.frame(logL=gr.op[[2]],n=1:length(gr.op[[2]])),
            aes(x=n,y=logL),
            color='blue')
```



2

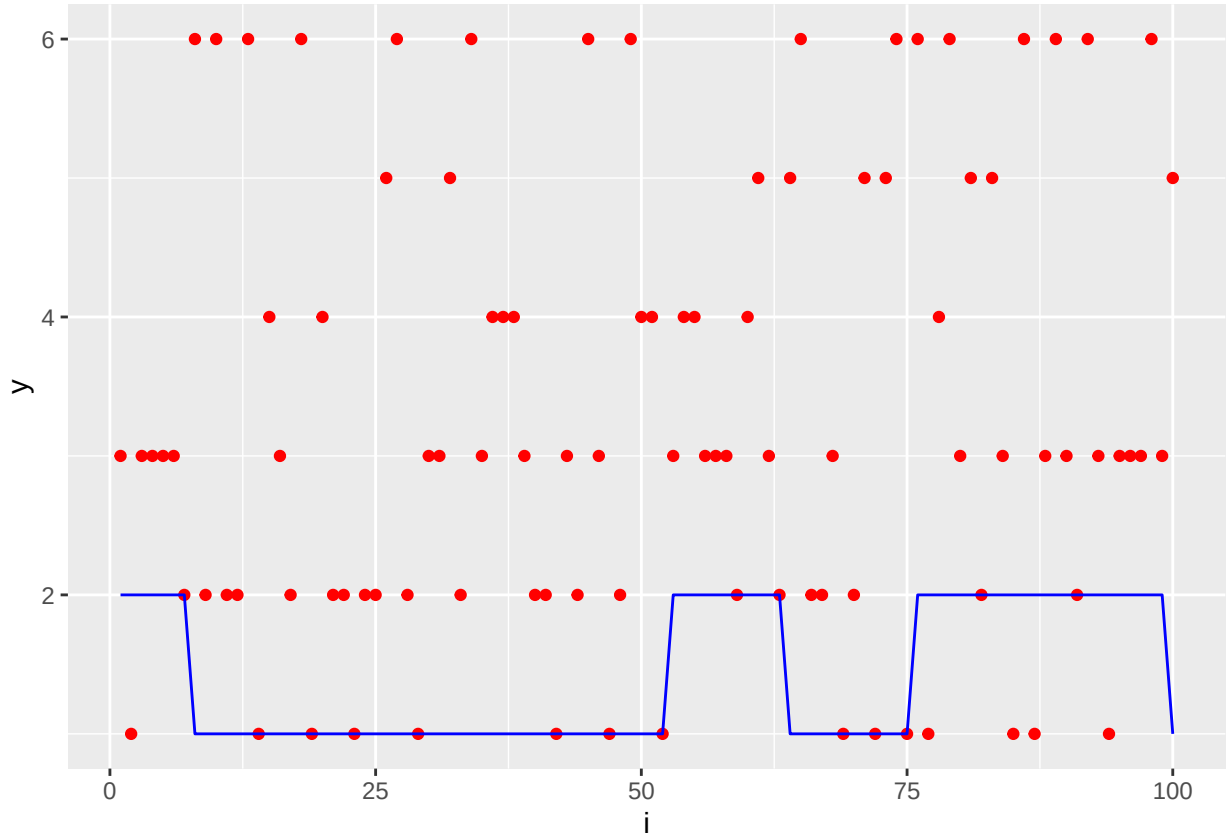
```
#' estimate parameters of EM algorithm
#' @param data a vector of data
#' @param init.value initial value for the vector p.
#' @return estimated parameters of p
#' @example EM(c(6,4,55,35),c(.1,.1,.8))
EM=function(data,init.value){
  na=data[1]
  nab=data[2]
  nb=data[3]
  no=data[4]
  n=na+nab+nb+no
  pa=init.value[1]
  pb=init.value[2]
  po=init.value[3]
  eps=1
  p.old=0
  p=1
  while( eps>0.001 & abs(p-p.old)>0.001){
    pa.old=pa
    pb.old=pb
    po.old=po
    p.old=p
    maa=na*(pa^2)/(pa^2+2*pa*po)
    mao=na*(2*pa*po)/(pa^2+2*pa*po)
    mbb=nb*(pb*pb)/(pb^2+2*pb*po)
    mbo=nb*(2*pb*po)/(pb^2+2*pb*po)
    mab=nab
    moo=no
    pa=(2*maa+mao+mab)/2/n
    pb=(2*mbb+mbo+mab)/2/n
    po=(2*moo+mao+mbo)/2/n
    eps=sqrt((pa-pa.old)^2+(pb-pb.old)^2+(po-po.old)^2)
    p=(pa^2+2*pa*po)^(na)*(pb^2+2*pb*po)^(nb)*(po^2)^(no)
  }
  return(list(c(pa=pa,pb=pb,po=po,p=p)))
}
```

#3

1

```
P=matrix(c(0.98,0.05,0.02,0.95),nrow=2)
E=matrix(c(1/6,1/10,1/6,1/10,1/6,1/2,1/6,1/10,1/6,1/10,1/6,1/10),nrow=2)
v=matrix(c(.5,.5),nrow=1)
size=100
S=matrix(rep(0,size*2),nrow=2)
S[1,1]=sample(1:2, 1, prob = v)
S[2,1]=sample(1:6, 1, prob = E[2,])
```

```
for( i in  2:size){
  S[1,i]=sample(1:2, 1, prob = P[S[1,i-1],])
  S[2,i]=sample(1:6, 1, prob = E[S[1,i],])
}
library(ggplot2)
plot.data=data.frame(cbind(1:ncol(S),t(S)))
colnames(plot.data)=c('i','x','y')
ggplot(data=plot.data,aes(x=i,y=y))+
geom_point(color='red')+
  geom_line(aes(x=i,y=x),color='blue')
```



2

forward

```
A=matrix(rep(0,2*size),nrow=2)
A[,1]=v*E[,S[2,1]]
for (i in 2:size){
  A[,i]=E[,S[2,i]]*(t(P)%*%A[,i-1])
}
p=sum(A[,size])
print(p)
```

```
## [1] 6.535366e-77
```

backward

```
B=matrix(rep(0,2*size),nrow=2)
B[,size]=c(1,1)
for (i in (size-1):1){
  B[,i]=P%*%(E[,S[2,i+1]]*B[,i+1])
}
p=sum(v*E[,S[2,1]]*B[,1])
print(p)
```

```
## [1] 6.535366e-77
```

##3

```
data=S
error=1
size=ncol(S)
num1=length(unique(S[1,]))
num2=length(unique(S[2,]))
A=matrix(rep(0,2*size),nrow=2)
B=matrix(rep(0,2*size),nrow=2)
G=array(0,dim=c(size,num1,num1))
gamma=array(0,dim=c(size,num1))
v=matrix(rep(1/num1,num1),ncol=num1)
P=matrix(rep(rep(1/num1,num1),2),ncol=num1)
E=matrix(rep(rep(1/num2,num1),num2),ncol=num2)

while (error>.1){
  theta.old=c(v,P,E)

  A[,1]=v*E[,S[2,1]]
  for (i in 2:size){
    A[,i]=E[,S[2,i]]*(t(P)%*%A[,i-1])
  }


  B[,size]=c(1,1)
  for (i in (size-1):1){
    B[,i]=P%*%(E[,S[2,i+1]]*B[,i+1])
  }
  #b0=sum(v*E[,S[2,1]]*B[,1])


  for (t in 2:size){
    for(i in 1:num1){
      for(j in 1:num1){
        G[t,i,j]=A[i,t-1]*P[i,j]*E[j,S[2,t]]*B[j,t]/sum(A[,t-1])
      }
    }
```

```
  }

  for (t in 1:size){
    for(i in 1:num1){
      gamma[t,i]=A[i,t]*B[i,t]/sum(A[,t]*B[,t])
    }
  }

  for (i in 1:num1){
    v[i]=gamma[1,i]
    for (j in 1:num1){
      P[i,j]=sum(G[-1,i,j])/sum(G[-1,i,])
    }
    for (m in 1:num2){
      E[i,m]= sum(gamma[,i]*(S[2,]==m))/sum(gamma[,i])
    }
  }
  error=sqrt(sum((c(v,P,E)-theta.old)^2))
}
print(list(v,P,E))
```

```
## [[1]]
##      [,1] [,2]
## [1,]  0.5  0.5
##
## [[2]]
##      [,1] [,2]
## [1,]  0.5  0.5
## [2,]  0.5  0.5
##
## [[3]]
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,] 0.15 0.22 0.27 0.11 0.09 0.16
## [2,] 0.15 0.22 0.27 0.11 0.09 0.16
```