Midterm Exam 2019

3/13/2019

Xinyi Lin

xl2836

Disease D is a chronic neurological condition that leads to fast deterioration of motor and cognitive functions and eventually leads to death. Based on a theoretical model, the survival time of a patient suffered from disease D very much depends on his or her disease onset age. To be specific, let as the survival time of a patient, and be the disease onset age, the conditional distribution for given is exponential with failure rate .

Since D is a chronic condition, its actual onset times are often unobserved. Suppose the disease onset ages in a population also follows an exponential with failure rate , where is an unknown parameter. Suppose are observed survival times of patients with disease D in a population. The public health researchers are interested in estimating the parameter in the population so that they could design disease prevention policies on target ages.

### 1. Write out the marginal distribution of , and the observed likelihood function of .

### 2. Design a univariate optimization algorithm (e.g. Golden search or Newton’s method) to find the MLE of the observed likelihood in (1), and specify each step of your algorithm. Implement the algorithm into an R function.

loglike = function(theta){  
 res = sum(log(0.01\*theta)-2\*log(0.01\*Y+theta))  
 return(res)  
}

golden\_max = function(func, a, b){  
 w = 0.618  
 theta0 = a+(b-a)\*(1-w)  
 theta1 = theta0 + (b-a)\*(1-w)\*w  
 tol = 1e-4  
 i = 0  
   
 #rlist = c("a", "b", "theta0", "theta1")  
 rlist = c(i, a, b, theta0, theta1)  
 while(abs(b-a)>tol){  
 i = i+1  
 if(func(theta1) > func(theta0)){  
 a=theta0;  
 theta0 = theta1  
 theta1 = theta0 + (b-a)\*(1-w)\*w  
 }  
 else{  
 b=theta1;  
 theta0 = a+(b-a)\*(1-w)  
 theta1 = theta0 + (b-a)\*(1-w)\*w  
 }   
 rlist = rbind(rlist, c(i, a, b, theta0, theta1))  
 }   
 #tail(rlist)  
 return(tail(rlist))  
}  
  
#golden\_max(fx, 0, 1.5)

### 3. Write out the joint distribution of , and design an EM algorithm to find the MLE of . Clealy write out the E-steps and M-steps in each iteration, and implement the algorithm into an R function.

# E-step evaluating conditional means E(Z\_i | X\_i , pars)  
# pars: parameters list  
delta <- function(X, theta){  
 return(1/(0.01\*x))  
}

# M-step - updating the parameters  
mles <- function(Z, X) {  
 n <- length(X)  
 thetahat <- n/sum(X)  
 return(thetahat)  
}

EMmix <- function(X, start, nreps=10) {  
 i <- 0  
 Z <- delta(X, start)  
 newpars <- start  
 res <- c(0, t(as.matrix(newpars)))  
 while(i < nreps) {  
 # This should actually check for convergence  
 i <- i + 1  
 newpars <- mles(Z, X)  
 Z <- delta(X, newpars)  
 res <- rbind(res, c(i, t(as.matrix(newpars))))  
 }  
 return(res)  
}

### 4.Simulate data sets with true , and apply the optimization functions you deleveped in (2) and (3) to estimate , which algoirhtm is more efficient (comparing the numbeter of iterations and computing times)?

n = 20  
true\_theta = 0.025  
X = rexp(n, true\_theta)  
Y = vector(mode = "numeric", n)  
for (i in 1:n) {  
 Y[i] = rexp(1, X[i])  
}  
X

## [1] 25.117039 51.566520 69.596758 53.201738 55.847084 40.613765  
## [7] 81.081494 30.158246 41.783432 99.823961 18.615312 32.958098  
## [13] 6.483670 103.554923 7.423856 30.452024 35.466654 40.321559  
## [19] 12.295472 89.796064

Y

## [1] 0.0319780107 0.0153454047 0.0044782434 0.0049256191 0.0007295083  
## [6] 0.0687709362 0.0051250296 0.0053896262 0.0306868024 0.0006270210  
## [11] 0.0007592391 0.0645179231 0.0197331507 0.0125721216 0.0592983738  
## [16] 0.0232106947 0.0053026736 0.0841198213 0.1396571449 0.0034313936

# method 1  
golden\_max(loglike, 0, 0.1)

## [,1] [,2] [,3] [,4] [,5]  
## 10 0.000000e+00 0.0008136147 3.108008e-04 0.0005028757  
## 11 0.000000e+00 0.0005028757 1.920985e-04 0.0003108154  
## 12 0.000000e+00 0.0003108154 1.187315e-04 0.0001921076  
## 13 0.000000e+00 0.0001921076 7.338509e-05 0.0001187371  
## 14 7.338509e-05 0.0001921076 1.187371e-04 0.0001467646  
## 15 7.338509e-05 0.0001467646 1.014161e-04 0.0001187392

### 5.Show that is the median of , and hence (the sample median of is a consistent estimation of as well.

### 6. Now that you have two estimates of , the MLE estimate and the one using the sample median of ’s, Carry out a simulation study to compare the estimation efficiency of the two estimates. Based on your simulation results, which estimate should be recommended?

compare <- function(n, p, N=10000) {  
 SSEmle <- SSEmedian <- 0  
 for(i in 1:N){  
 SSEmle <- SSEmle + mean(Y)^2  
 SSEmedian <- SSEmedian + median(Y)^2  
 }  
 return(list(n=n, p=p, MSEmean = SSEmle / N,  
 MSEmedian = SSEmedian / N))  
}

pvec <- seq(0, 0.3, by=0.025)  
res <- NULL  
for(i in 1:length(pvec))  
 res <- rbind(res, as.numeric(compare(20, pvec[i], N=5000)))  
res <- data.frame(res)  
names(res) <- c("n", "p", "MSEmean", "MSEmedian")  
print(res)