Lab6

Andreas C Charitos(andch552) Omkar Bhutra (omkbh878)

8 March 2019

Question 1 -Genetic Algorithm

Subquestion 1-Define target function

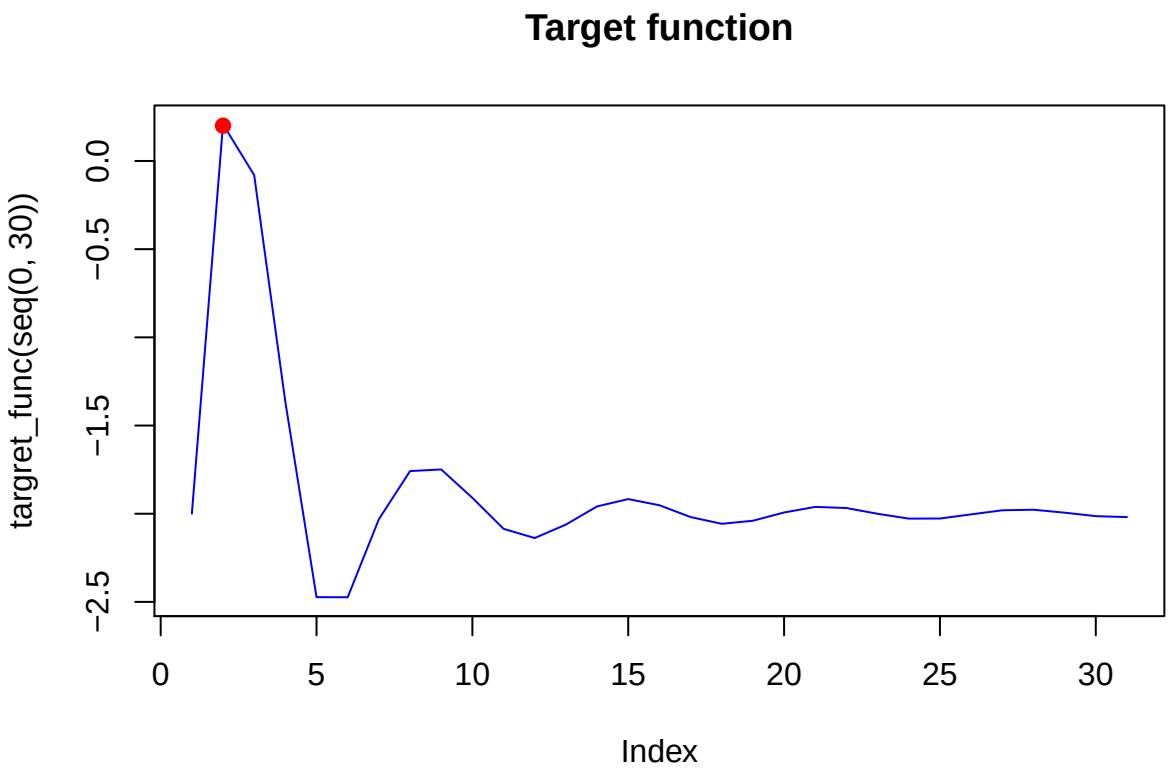
We are going to perform one-dimensinal maximization with the help of genetic algorithm for the following function :

$$f(x) = \frac{x^2}{e^x} - 2e^{(-\frac{(9sin(x))}{(x^2+x+1)})}$$

```
library(animation)
```

```
## Warning: package 'animation' was built under R version 3.5.2
```

```
#our target function we wish to
#find the maximum

targret_func<-function(x){

  result1<-x^2/exp(x)
  result2<-2*exp( -(9*sin(x)) / (x^2+x+1) )

  return(result1-result2)

}


plot(targret_func(seq(0,30)),type = "l",col="blue",
     main="Target function")
points(2,0.2,col="red",pch=19)
```



In the plot we can see that the function has some local maximum but with red circle is the global maximum we wish to approximate using genetic algorithm.

Subquestion 2-Define crossover function

We are asked to perform the crossover function that for 2 scalars we get their "kid"

```
#crossover function ,given 2 integers
#returns their kid
crossover<-function(x,y){
  kid<-(x+y)/2
  return(kid)
}
```

Subquestion 3-Define mutation function

The function mutate() takes a scalar x returns the result of the integer division $x^2 mod\ 30$ we will use this function to perform a mutation to the "kid" that will be produced in the crossover.

```
#mutate function
#takes a scalar and performs
#mutation
mutate<-function(x){
  result<-x^2 %% 30

  return(result)
```

```
}
```

Subquestion 4-Define genetic function

we are going to implement the genetic function that has 2 arguments number of iterations and mutation probability

```
genX<-function(maxiter,mutprob,animated=F){
  #plot funciton that plots the original
  #and the final population obtained
  plotf<-function(){
    plot(targret_func(seq(0,30)),type = "l",col="blue",lwd=2,
         main= paste0("maxiter=",maxiter," and mutprob=",mutprob),
         ylab="Value")
    points(initial_pop,targret_func(initial_pop),col="red",
           pch=19)
    return()

  }


  #initialize a population
  initial_pop=seq(0,30,5)
  #compute the value from target function
  #for each population
  Values<-sapply(initial_pop,targret_func)
  #initialize vector to store
  #max value for every iteration
  max_Value<-rep(0,maxiter)
  #if statement if we want gif animation
  if (animated==F){
    for (i in 1:maxiter){
      #find 2 parents from the initial population
      parents<-initial_pop[c(sample(1:length(initial_pop),2,replace = F))]
      #get the victim that is going to be replaced
      victim<-order(Values)[1]
      #take "kid" from 2 parents with crossover
      new_kid<-crossover(parents[1],parents[2])
      #check the probability for mutation
      #if is higher than mutprob "kid"
      #will be mutated
      if (runif(1)<mutprob){
        new_kid<-mutate(new_kid)
      }
      #change the kid with the victim
      #in the initial population
      initial_pop[victim]<-new_kid
      #calculate the values of the new
      #population again
      Values<-sapply(initial_pop,targret_func)
      #get the max value
      max_Value[i]<-max(Values)
```

```
    }
    #plotf()
    return(list("plot"=plotf(),"final_pop"=max_Value))

  }
  #use this is you want animation gif
  else{
    require(animation)
    saveGIF({
  ani.options(interval=.3)
  col.range <- heat.colors(15)

  for (i in 1:maxiter){

    parents<-initial_pop[c(sample(1:length(initial_pop),2,replace = F))]
    victim<-order(Values)[1]
    new_kid<-crossover(parents[1],parents[2])

    if (runif(1)<mutprob){
      new_kid<-mutate(new_kid)
    }

    initial_pop[victim]<-new_kid
    Values<-sapply(initial_pop,targret_func)
    max_Value[i]<-max(Values)

    plot(targret_func(seq(0,30)),type = "l",col="blue",main=paste0("plot for",i,"th iteration"))
    points(initial_pop,targret_func(initial_pop),col="red")

    }#end of for loop

    })#end of animation
  }


}
```

Subquestion 5-Run the function with diffrent settings

Last we are goning to try the algorithm for different combinations of maxiter= 10, 100 and mutprob= 0.1, 0.5, 0.9

```
set.seed(1234567)

par(mfrow=c(3,3))
#maxiter=10
genX(10,0.1,animated=F)$plotf
```

```
## NULL
```

```
genX(10,0.5,animated=F)$plotf
```

```
## NULL
```

```
genX(10,0.9,animated=F)$plotf
```

```
## NULL
```

```
#maxiter=100
genX(100,0.1,animated=F)$plotf
```

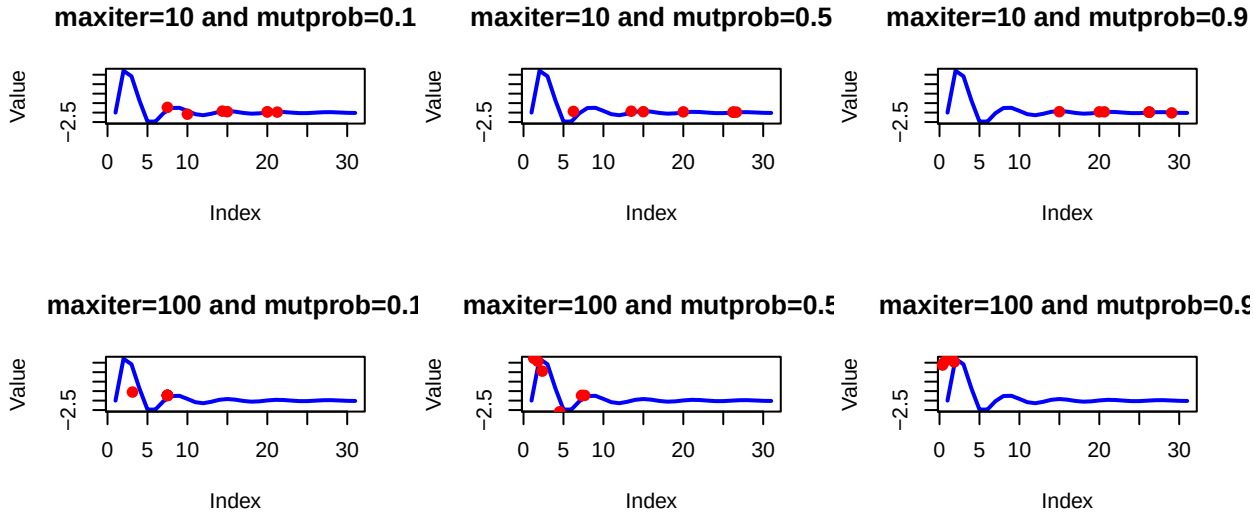
```
## NULL
```

```
genX(100,0.5,animated=F)$plotf
```

```
## NULL
```

```
genX(100,0.9,animated=F)$plotf
```

```
## NULL
```

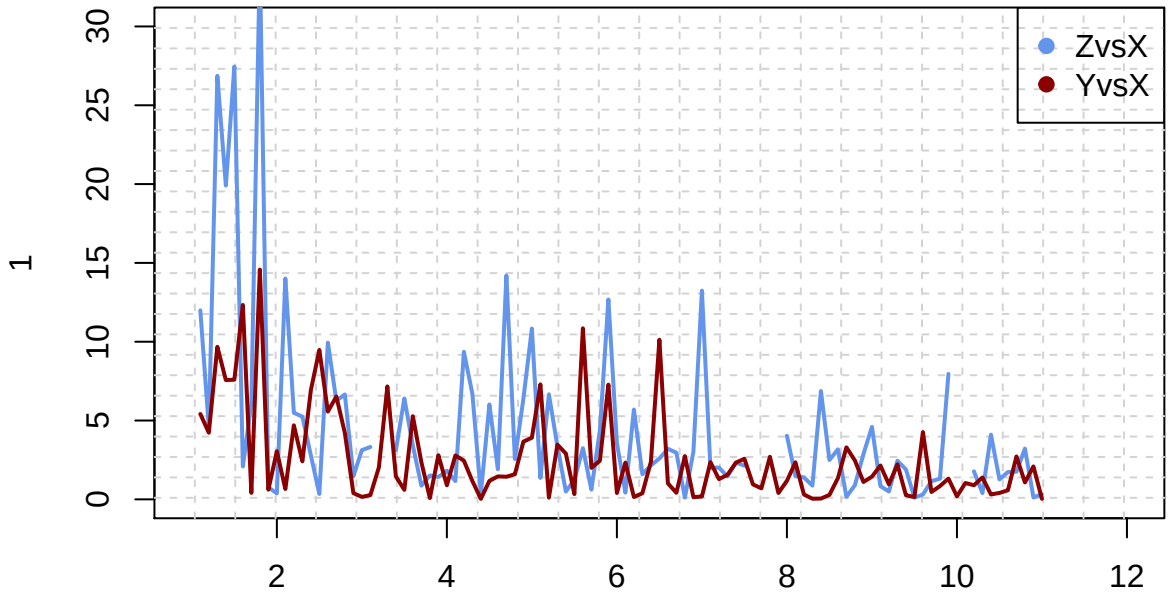


Finaly, the plot shows that with maxiter=100 have better performance than the one with maxiter=10 and the final population values are more closer to maximum. This is reasonable because the algorithm needs more iterations in order to make more crossovers and mutations.Furthermore,if the mutation probability is high the mutatation is very ofter for the "kids"-points and that makes them change points and explore new points which is the purpose of the mutation.

Question 2 -EM Algorithm

Subquestion 1-Run the dependence of Z,Y Vs X

```
phys<-read.csv("physical1.csv",sep=",")
plot(1, type="n", xlab="",ylim = c(0,30),xlim=c(1,12))
grid(25,25,lty=2)
lines(phys$X,phys$Z,type="l",col="cornflowerblue",
     lwd=2,ylab = "Z,Y",xlab="X",
     main="Z vs X")
lines(phys$X,phys$Y,col="darkred",lwd=2)
legend("topright",legend=c("ZvsX","YvsX"),col=c("cornflowerblue","darkred"),
       pch=19)
```



Yes, it seems that the two responses are related to each other. It is observed that both responses Y and Z have decayed oscillations, this makes it seem that the responses are related to each other. The decay in response Z is lower than in Y.

Subquestion 2-Derive EM algorithm for lambda

If we assume that the data Y and Z are independent and for a sample size n the likelihood is given by :

$$L(\lambda|Y,Z) = \prod_{i=1}^{n} f(Y).\prod_{i=1}^{n} f(Z)$$

$$= \prod_{i=1}^{n} \frac{X_i}{\lambda}.e^{\frac{-X_i.Y_i}{\lambda}} . \prod_{i=1}^{n} \frac{X_i}{2\lambda}.e^{\frac{-X_i.Z_i}{2\lambda}}$$

but some values of Z are unobserved if we denote r the size of observed data in Z then we have :

$$\prod_{i=1}^{n} \frac{X_i}{\lambda}.e^{\frac{-X_i.Y_i}{\lambda}} . \prod_{i=1}^{r} \frac{X_i}{2\lambda}.e^{\frac{-X_i.Z_i}{2\lambda}} . \prod_{i=r+1}^{n} \frac{X_i}{2\lambda}.e^{\frac{-X_i.Z_i}{2\lambda}}$$

The log-liklihood is then given by :

$$logL = \sum_{i=1}^{n} logX_i - nlog\lambda - \frac{1}{\lambda}\sum_{i=1}^{n} X_iY_i + \sum_{i=i}^{r} logX_i - rlog(2\lambda) - \frac{1}{\lambda}\sum_{i=1}^{r} \frac{X_iZ_i}{2} + \sum_{i=r+1}^{n} logX_i - (n-r)log(2\lambda) - \frac{1}{\lambda}\sum_{i=r+1}^{n} \frac{X_iZ_i}{2} (1)$$

E-step:

$$Q(\lambda, \lambda_k) = E[L(\lambda/k, Y, Z)/\lambda_k, Y] =^{(1)}$$

$$2\sum_{i=1}^{n} logX_i - nlog\lambda - nlog(2\lambda) - \frac{1}{\lambda}\sum_{i=i}^{r} \frac{X_iY_i}{2} - \frac{1}{\lambda}\sum_{i=1}^{r} \frac{X_iZ_i}{2} - \sum_{i=r+1}^{n} \frac{X_i}{2\lambda} E[Z_i] =$$

$$2\sum_{i=1}^{n} logX_i - nlog\lambda - nlog(2\lambda) - \frac{1}{\lambda}\sum_{i=i}^{r} \frac{X_iY_i}{2} - \frac{1}{\lambda}\sum_{i=1}^{r} \frac{X_iZ_i}{2} - \sum_{i=r+1}^{n} \frac{X_i}{2\lambda} \frac{2\lambda_k}{X_i} =$$

$$2\sum_{i=1}^{n} logX_i - nlog\lambda - nlog(2\lambda) - \frac{1}{\lambda}\sum_{i=i}^{r} \frac{X_iY_i}{2} - \frac{1}{\lambda}\sum_{i=1}^{r} \frac{X_iZ_i}{2} - (n-r)\frac{\lambda_k}{\lambda}$$

M-step: The maximum likelihood estimate of the parameters is obtained by taking the partial derivative with respect to λ. This is repeated till the solution converges.

$$\frac{\partial L(\lambda|Y, Z)}{\partial \lambda} = 0$$

$$\lambda = \frac{1}{2n}(\sum_{i=1}^{n} X_iY_i + \sum_{i=1}^{r} \frac{X_iZ_i}{2} + (n-r)\lambda_k)$$

Subquestion 3-Implement the algorithm

We are now going to implement the algorithm with $\lambda_0 = 100$ and stoping criterion if change in λ is less than 0.001

```
library(knitr)

EM <- function(data, eps, kmax=500, lambda){
    X<-phys$X
    Y<-phys$Y
    Z<-phys$Z

    Xobs <- X[!is.na(Z)]
    Zobs <- Z[!is.na(Z)]
    Zmiss <- Z[is.na(Z)]

    n <- length(X)
    r <- length(Zobs)
```

```
  floglik<-function(l,l_k){

  L=2*sum(log(X))-n*log(l)-(X%*%Y)/l-n*log(2*l)-(Xobs%*%Zobs)/(2*l)-(n-r)*(l_k/l)
  return(L)

  }


  #initial state
  k <- 0
  lambda_prev <- lambda*10+10 # make some difference
  lambda_curr <- lambda
  loglike <- floglik(lambda_curr,lambda_prev)
  print(loglike)
  df<-NULL
  df<-rbind(df,c(k ,lambda_curr,loglike))
  colnames(df)<-(c("iter","lambda","loglikelihood"))

  while ((abs(lambda_prev-lambda_curr)>eps) && (k<(kmax+1))){
    lambda_prev <- lambda_curr
    # since we already know how to get argmax lda
    lambda_curr <- 1/(2*n)*(X%*%Y+Xobs%*%Zobs/2+(n-r)*lambda_prev)
    k<-k+1
    loglike <- floglik(lambda_curr,lambda_prev)
    df <- rbind(df,c(k, lambda_curr,loglike))

    #pretty table
    print(kable(df))

    }
return(lambda_curr)
}
```

```
lda_opt <- EM(phys, eps=0.001, lambda=100)
```

```
##           [,1]
## [1,] -761.7647
##
##
##  iter      lambda   loglikelihood
## -----  ----------  --------------
##     0   100.00000       -761.7647
##     1    14.26782       -470.9963
##
##
##  iter      lambda   loglikelihood
## -----  ----------  --------------
##     0   100.00000       -761.7647
##     1    14.26782       -470.9963
##     2    10.83853       -416.0165
##
##
##  iter      lambda   loglikelihood
```

```
## -----  ----------  --------------
##     0   100.00000       -761.7647
##     1    14.26782       -470.9963
##     2    10.83853       -416.0165
##     3    10.70136       -413.4692
##
##
##  iter      lambda   loglikelihood
## -----  ----------  --------------
##     0   100.00000       -761.7647
##     1    14.26782       -470.9963
##     2    10.83853       -416.0165
##     3    10.70136       -413.4692
##     4    10.69587       -413.3666
##
##
##  iter      lambda   loglikelihood
## -----  ----------  --------------
##     0   100.00000       -761.7647
##     1    14.26782       -470.9963
##     2    10.83853       -416.0165
##     3    10.70136       -413.4692
##     4    10.69587       -413.3666
##     5    10.69566       -413.3625
```

This result indicates that the optimal lambda value is 15.96223 after 6 iterations that are needed for convergence.

Subquestion 4-Plot E[Y] E[Z]

The plot above shows the E[Y] E[Z] versus X in the same plot as Y and Z vs X

From the graph , it is observed that both exponential's of Y and Z capture the flow of Y and Z against X. Hence, the lambda found from the above algorithm is optimal.