Computational Statistics Lab 6

Vasileia Kampouraki, Weng Hang Wong, Jooyoung Lee

28/2/2020

Question 1: Genetic algorithm

In this assignment, you will try to perform one-dimensional maximization with the help of a genetic algorithm.

1. Define the function

$$f(x) := \frac{x^2}{e^x} - 2exp(-(9sinx)/(x^2 + x + 1))$$

```
f <- function(x){
  f= x^2/exp(x) - 2*exp(-(9*sin(x))/(x^2 + x + 1))
  return(f)
}
```

2. Define the function crossover(): for two scalars x and y it returns their "kid" as (x+y)/2.

```
crossover <- function(x,y){
  kid = (x+y)/2
  return(kid)
}
```

3. Define the function mutate() that for a scalar x returns the result of the integer division x^2 mod 30. (Operation mod is denoted in R as %%).

```
mutate <- function(x){
  res = x^2 %% 30
  return(res)
}
```

4. Write a function that depends on the parameters *maxiter* and *mutprob* and:

(a) Plots function f in the range from 0 to 30. Do you see any maximum value?

(b) Defines an initial population for the genetic algorithm as X = (0,5,10,15,. . . ,30).

(c) Computes vector Values that contains the function values for each population point.

(d) Performs maxiter iterations where at each iteration

i. Two indexes are randomly sampled from the current population, they are further used as parents (use sample()).

ii. One index with the smallest objective function is selected from the current population,the point is referred to as victim (use order()).

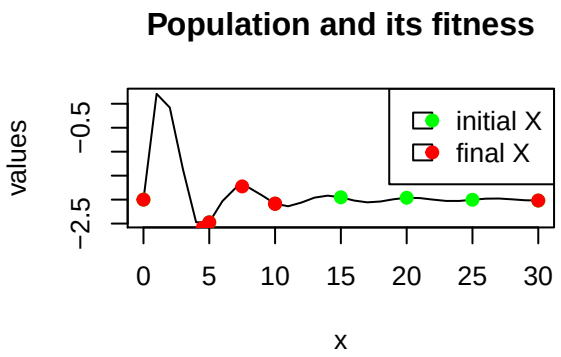
iii. Parents are used to produce a new kid by crossover. Mutate this kid with probability mutprob (use crossover(), mutate()).

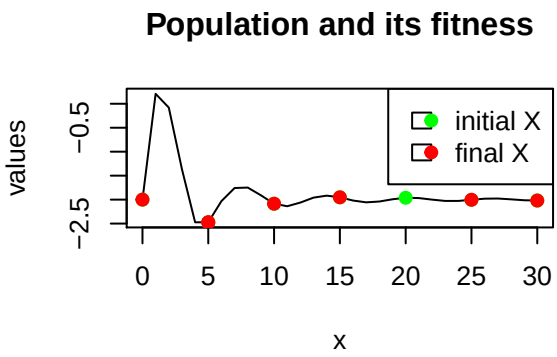
iv. The victim is replaced by the kid in the population and the vector Values is updated.

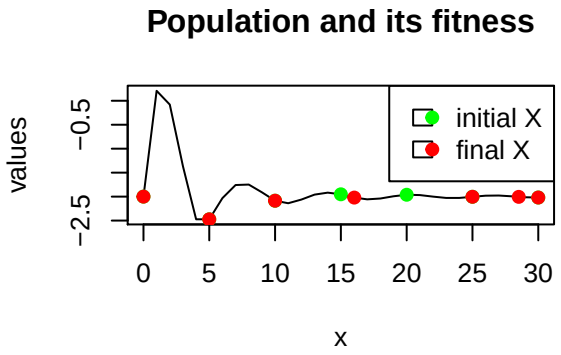
v. The current maximal value of the objective function is saved.

(e) Add the final observations to the current plot in another colour.

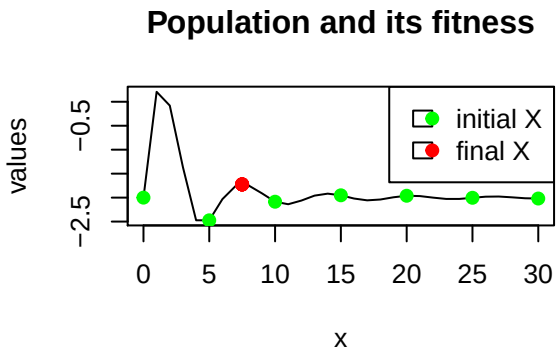
```
## The max value of the initial population is

## [1] -1.951947

## $population
## [1]  0.000000  5.000000 10.000000 10.000000  4.501953  7.500000 30.000000
##
## $values
## [1] -2.000000 -2.473573 -2.085654 -2.085654 -2.589527 -1.724415 -2.019194
##
## $max_value
## [1] -1.724415

## $population
## [1]  0  5 10  5 25 15 30
##
## $values
## [1] -2.000000 -2.473573 -2.085654 -2.473573 -2.003663 -1.951947 -2.019194
##
## $max_value
## [1] -1.951947

## $population
## [1]  0.00000  5.00000 10.00000 28.51562 16.01959 25.00000 30.00000
##
## $values
## [1] -2.000000 -2.473573 -2.085654 -2.005111 -2.020242 -2.003663 -2.019194
##
## $max_value
## [1] -2
```

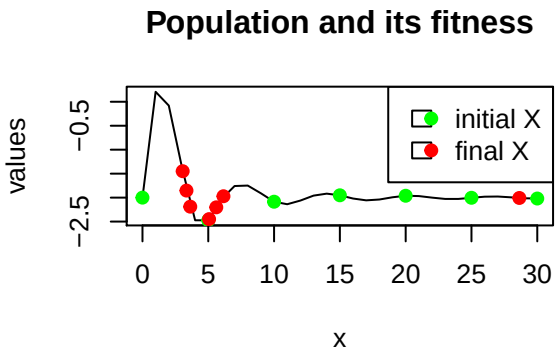


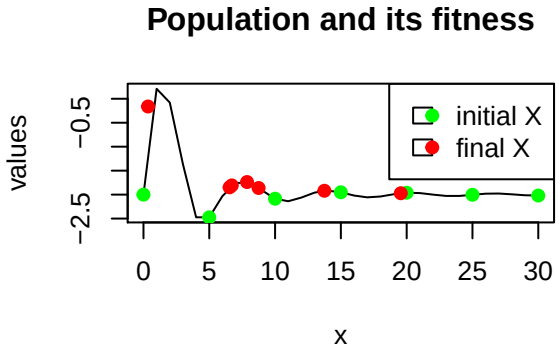





```
## $population
## [1] 7.491455 7.491455 7.491455 7.495728 7.492256 7.495193 7.500000
##
## $values
## [1] -1.724469 -1.724469 -1.724469 -1.724440 -1.724463 -1.724443 -1.724415
##
## $max_value
## [1] -1.724415

## $population
## [1]  3.055912  6.155119  5.063112  3.632799  3.344356 28.639149  5.609116
##
## $values
## [1] -1.448585 -1.971282 -2.448986 -2.188693 -1.852956 -2.007571 -2.202702
##
## $max_value
## [1] -1.448585

## $population
## [1] 19.5532038  6.5468688  6.7123415 13.7480012  7.8706691  8.7506093  0.3485509
##
## $values
## [1] -1.9712997 -1.8475734 -1.8082006 -1.9197099 -1.7376965 -1.8618524 -0.1613961
##
## $max_value
## [1] -0.1613961
```







Genetic algorithm is widely used for optimization problems. This algorithm is inspired by the evolutionary theory and the ideas of natural selection.

So, based on this idea, we start with an initial population and based on an objective function that evaluates the fitness of each point in the population we keep every time those candidates that have the best fitness values and then those candidates reproduce and give the offsprings. This is done using the successor functions, which are *crossover()* and *mutation()* and then comes selection.

Crossover is used to combine two parents and produce a child and *mutation* takes a candidate and returns a slightly different candidate.

Mutation is very important for the genetic algorithm because sometimes some important genes might be missing from the population and thus, mutation allows to search as much as possible in the search space and avoid getting stuck in local maxima(or minima).

At first we are asked to plot function f in the range from 0 to 30 and see if there's a maximum value. The maximum value we get from f for the initial population X=(0,5,...,30) is -1.951947.

The first three plots represent the results of the algorithm run with **maxiter**=10 and **mutprob**=0.1,0.5 and 0.9 respectively. It is possible to observe that the final `X` are wider spread as the value of **mutprob** increases.

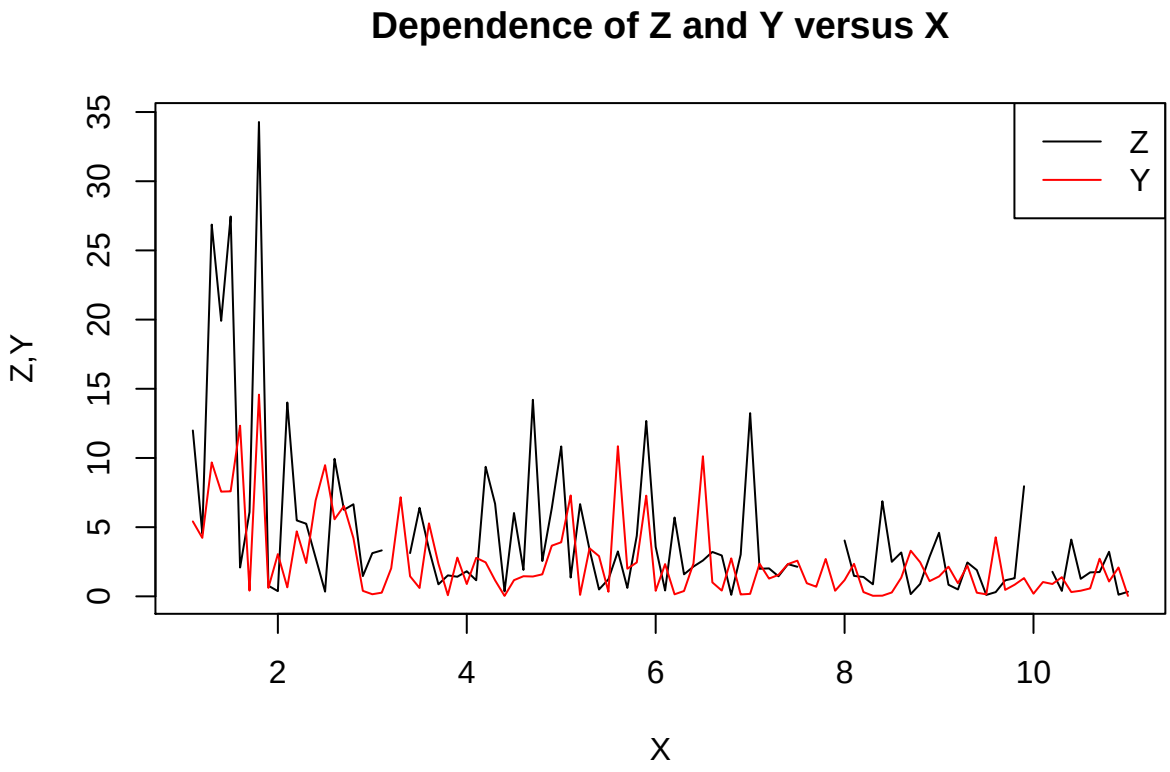
The last three plots represent the results of the algorithm run with **maxiter**=100 and **mutprob**=0.1,0.5 and 0.9 respectively. Compare to the previous graphs with **maxiter**=10, it is possible to observe that the final `X` of the same **mutprob** are rather concentrated. Unlike **maxiter**=10, having higher mutation probability does not lead to wider spread final `X`. This is because although the higher mutation probability leads to wider search over population, repeating of breeding leads to convergence of the final population.

Question 2: EM algorithm

The data file physical.csv describes a behavior of two related physical processes Y=Y(X) and Z=Z(X).

1. Make a time series plot describing dependence of Z and Y versus X. Does it seem that two processes are related to each other? What can you say about the variation of the response values with respect to X?

```
data <- read.csv("~/Computational Stat/732A90_VT2020_Materials/physical1.csv")
X <- data$X
Y <- data$Y
Z <- data$Z
```



First thing we observe when looking at this time series plot is that there Z has some blank points which means that some data are missing. Z is has quite a few peaks, with the highest ones being observed for small values of X. Y also has a quite similar pattern to Z, with peaks that are more intense at the beginning, but generally smaller values than Z. As we said, the variation of both response values compared to X is higher for small values of X and smaller for bigger values of X.

2. Note that there are some missing values of Z in the data which implies problems in estimating models by maximum likelihood. Use the following model $Y_i \sim exp(X_i/\lambda)$, $Z_i \sim exp(X_i/2\lambda)$ where λ is some unknown parameter. The goal is to derive an EM algorithm that estimates λ.

3. Implement this algorithm in R, use $\lambda_0 = 100$ and convergence criterion "stop if the change in λ is less than 0.001". What is the optimal λ and how many iterations were required to compute it?

With optimal value λ, loglikelihood will be maximized. However, due to several missing values of `Z`, expected value of loglikelihood will be maximized instead. Through this task, we will try to estimate the optimal value λ that could return the maximum expected value of loglikelihood. Furthermore, the number of iterations performed to obtain optimal λ will be found.

The process of obtaining above mentioned two is based on the following mathematics:

1. Assume that Y and Z are independent.
2. Calculate likelihood function of λ

$$L(\lambda|Y,Z) = \frac{\prod_{i=1}^{n} x_i^2}{(2\lambda^2)^n}\ e^{-\frac{\sum_{i=1}^{n} x_i y_i}{\lambda} - \frac{\sum_{i=1}^{n} x_i z_i}{2\lambda}}$$

3. Calculate loglikelihood function of λ

$$l(\lambda|Y,Z) = -2nlog(\lambda) - \frac{\sum_{i=1}^{n} x_i y_i}{\lambda} - \frac{\sum_{i=1}^{n} x_i z_i}{2\lambda} + C$$

The terms that does not contain λ do not affect loglikelihood value, because those values are fixed. Therefore, on above mathematical expression, such terms are written as a constant, C.

4. Find expected loglikelihood

$$E[l(\lambda|Y,Z)] = E[\ -2nlog(\lambda) - \frac{\sum_{i=1}^{n} x_i y_i}{\lambda} - \frac{\sum_{i \in observed} x_i z_i}{2\lambda} - \frac{\sum_{j \in unobserved} x_j z_j}{2\lambda} + C\ ]$$

All terms except $\frac{\sum_{j \in unobserved} x_j z_j}{2\lambda}$ are from observed data. Expected value of a constant number is a constant number. So there is only one term that involves uncertainty. Since `Z` follows exponential distribution with parameter $\frac{x_j}{2\lambda_{t-1}}$, unobzerved Z_i's expected value will be $\frac{2\lambda_{t-1}}{x_j}$, such that λ_{t-1} represents λ value obatained in previous iteration.

$E[\ \frac{\sum_{j \in unobserved} x_j z_j}{2\lambda}\ ] = \sum_{j \in unobserved} \frac{x_j}{2\lambda} \frac{2\lambda_{t-1}}{x_j} = \sum_{j \in unobserved} \frac{\lambda_{t-1}}{\lambda} = U\ \frac{\lambda_{t-1}}{\lambda}$, U is a number of unobserved `Z`.

5. Take a derivative of the expected loglikelihood with respect to λ

$$\frac{\partial l(\lambda)}{\partial \lambda} = \frac{-2n}{\lambda} + \frac{\sum_{i=1}^{n} x_i y_i}{\lambda^2} + \frac{\sum_{i \in observed} x_i z_i}{2\lambda^2} + U\ \frac{\lambda_{t-1}}{\lambda^2}$$

6. Set above equal to 0, find λ and update its value

$$\lambda = \frac{(\sum_{i=1}^{n} x_i y_i) + (\frac{\sum_{i \in observed} x_i z_i}{2}) + (U\lambda_{t-1})}{2n}$$

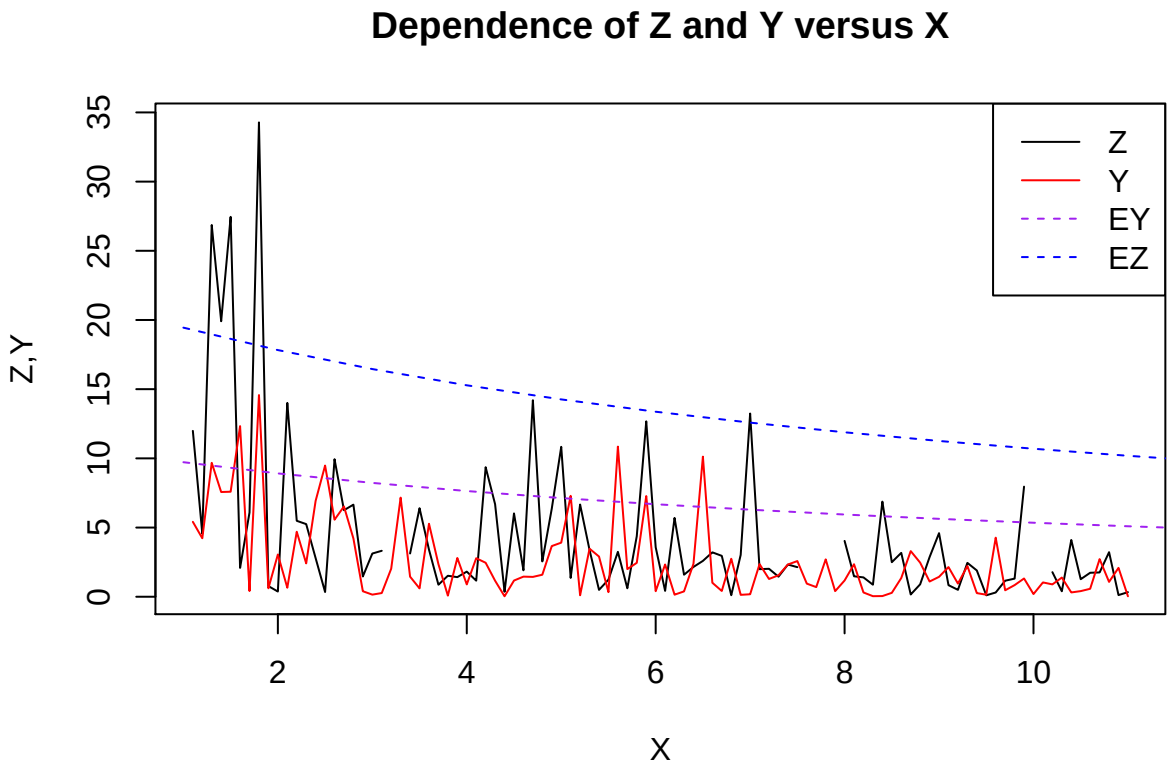
```
## The optimal lambda is:
```

```
## [1] 10.69566
```

```
## and the number of iterations required is:
```

```
## [1] 6
```

4. Plot E[Y] and E[Z] versus X in the same plot as Y and Z versus X. Comment whether the computed λ seems to be reasonable.



Since $Y_i \sim exp(X_i/\lambda)$, $Z_i \sim exp(X_i/2\lambda)$ the expected value of Y will be E(Y)= λ/X and the expected value of Z will be E(Z)= λ/2X.

Both lines that represent the expected values seen in the plot seem to be reasonable for representing the mean as they have more or less the same pattern as the data so it seems that the mean is following. Also, E(Y) is higher in the plot and this is actually reasonable as Z has generally a bit higher values compared to Y and also the mean of Z is two times the mean of Y.

Appendix

```
f <- function(x){
  f= x^2/exp(x) - 2*exp(-(9*sin(x))/(x^2 + x + 1))
  return(f)
}
```

```r
crossover <- function(x,y){
  kid = (x+y)/2
  return(kid)
}
mutate <- function(x){
  res = x^2 %% 30
  return(res)
}
set.seed(12345)
GA <- function(maxiter,mutprob){

  X <- seq(0,30,5)
  v <- c(0:30)
  initialplot <-  plot(v,f(v),main="Population and its fitness",xlab="x",ylab = "values",type="l")
  points(X,f(X),col="green",type = "p",pch=19) #plot also the initial population for comparison


  #values <- vector(length=length(X))
  parent <- vector(length = 2)
  victim <- 0
  max_value <- 0

  X <- seq(0,30,5) #initialize population
  values <- f(X)

  i=1
  while (i <= maxiter){

      parent <- sample(X,2)

      victim <- which.min(order(values)) #index of the smallest obj func value

      kid <- crossover(parent[1],parent[2])

      #Use runif to see if new kid needs to be mutated
      prop_mutat = runif(1,0,1)

      if (prop_mutat < mutprob){
        new_kid = mutate(kid)
      } else {
        new_kid = kid
      }

      #replace victim by new kid and update values
      X[victim] = new_kid
      values[victim] = f(new_kid)


      max_value = max(values) #current maximal value of the objective function
      i=i+1

  }
  initialplot
```

```
  points(X,values,col="red",type = "p",pch=19)
  legend("topright",-1, -0.5, legend=c("initial X", "final X"),
       col=c("green", "red"), pch=19)
  return(list("population"=X,"values"=values,"max_value"=max_value))
}

X <- seq(0,30,5)
cat("The max value of the initial population is","\n")
max(f(X))

par(mfrow=c(2,2))
GA(10,0.1)
GA(10,0.5)
GA(10,0.9)

par(mfrow=c(2,2))
GA(100,0.1)
GA(100,0.5)
GA(100,0.9)

data <- read.csv("~/Computational Stat/732A90_VT2020_Materials/physical1.csv")
X <- data$X
Y <- data$Y
Z <- data$Z
plot(X,Z,main="Dependence of Z and Y versus X",type="l",ylab="Z,Y")
lines(X,Y,col="red")
legend("topright",legend=c("Z", "Y"),lty =1,
       col=c("black", "red"))
set.seed(12345)

n= nrow(data)
Zobs= Z[!is.na(Z)]
Zmiss= Z[is.na(Z)]

M= length(Zmiss)

lambda <- vector()
lambda[1] = 100

i=2
repeat{
  lambda[i] = (sum(X*Y) + 0.5*sum(X[which(!is.na(Z))]*Zobs) + M*lambda[i-1]) / (2*n)

  if (abs(lambda[i]- lambda[i-1]) < 0.001){
    break()
  }
  i = i+1

}

num_iter = length(lambda)
optimal = lambda[num_iter]
```

```r
cat("The optimal lambda is:","\n")
optimal

cat("and the number of iterations required is:","\n")
num_iter

EY <- optimal/X
EZ <- (2*optimal)/X
plot(X,Z,main="Dependence of Z and Y versus X",type="l",ylab="Z,Y")
lines(X,Y,col="red")
lines(EY,col="purple",lty=2)
lines(EZ,col="blue",lty=2)


legend("topright",legend=c("Z", "Y","EY","EZ"),lty =c(1,1,2,2),
       col=c("black", "red","purple","blue"))
```