732A90: Computational Statistics

Computer lab6 - Group11

Sofie Jörgensen, Oriol Garrobé Guilera, David Hrabovszki

06 March 2020

Question 1: Genetic algorithm

In this exercise we are going to perform one-dimensional maximization by using a genetic algorithm.

1.

Firstly, we define the function `f()` as

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

2.

Secondly, we define the function `crossover()`, that takes two scalars x and y as inputs, and returns a child as $\frac{x+y}{2}$.

3.

Thirdly, we define the function `mutate()`, that performs the integer division x^2 mod 30, for a scalar input x.

4.

Further, we will create a function called `genetic()`, with the parameters `maxiter` and `mutprob`. The settings of this `genetic()` function, as well as its output results, are presented in (a)-(e). The code can be found in the Appendix.

(a) The function $f()$ is plotted in the range from 0 to 30 in Figure X, and we can observe that there is a maximum value located at $x = 1.2$ and the maximum value is 0.2341 (with step size = 0.1). This maximum location is indicated by the blue vertical line.

(b) An initial population for the genetic algorithm is defined as $X = (0, 5, 10, 15, ..., 30)$.

(c) A vector called `Values` are computed, containing the function values for each population point.

(d) The `genetic()` function performs `maxiter` iterations. For each iteration:

i. Two parent indexes are randomly sampled from the current population.

ii. A victim index is selected from the current population, which has the smallest objective function value.

iii. A new child is produced from parents by `crossover()`. The new child is `mutate()`-d with probability `mutprob`.

iv. The victim is replaced by the new child in the population and `Values` is updated.

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

(e) The final observations are added to the current plot as red points.

5.

By using the defined functions from previous tasks (1.1-1.4), we are going to observe the initial population and final population. The final population is calculated by running the code with different combinations of `maxiter`$= 10, 100$ and `mutprob`$= 0.1, 0.5, 0.9$. Figure 1 plots `f(x)` and its maximum location is indicated by the blue vertical line. The initial population is depicted with the red points.

We can observe what the final populations look like after the runs with different combinations of parameters in figure 2. The table below gives information about the achieved maximum value of the final populations. It is clear that after just 10 iterations the lowest point in the population (at x=5) was replaced by a higher value. Even between the 3 populations that ran for 10 iterations, there are differences. This is because if the newly generated child gets mutated with a higher probability, its value will vary more. After 100 iterations however, all 3 populations seemed to "find" the maximum value of the objective function.

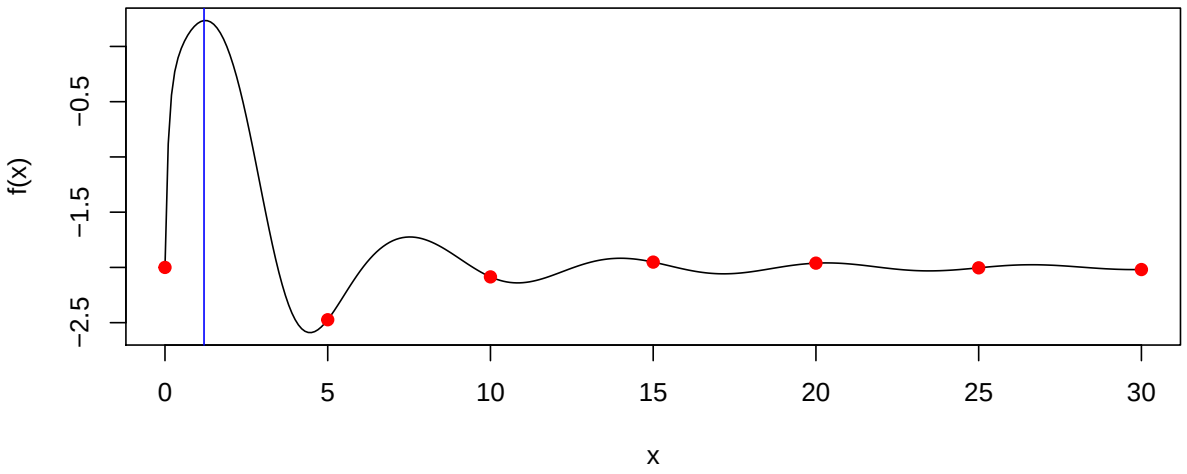

Figure 1: Initial population (red points)

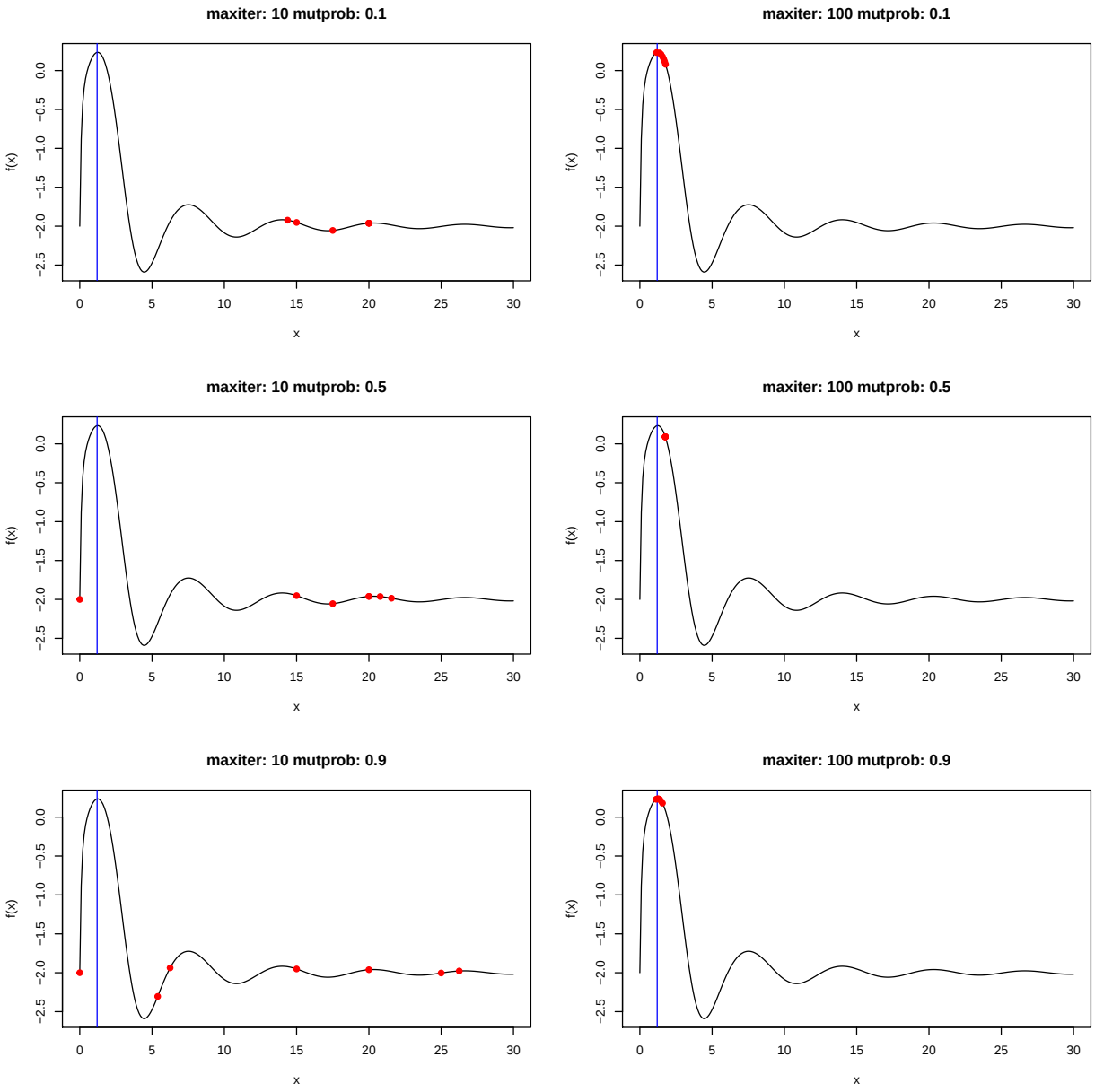

Figure 2: Genetic algorithm maximization

Table 1: Maximum values

	maxiter 10	maxiter 100
mutprob 0.1	-1.922621	0.2315820
mutprob 0.5	-1.951947	0.0901318
mutprob 0.9	-1.937529	0.2348200

Question 2: EM algorithm

The purpose with this exercise is to implement the EM algorithm. For this, we are given the data file `physical1.csv`, containing a behavior of two related physical processes $Y = Y(X)$ and $Z = Z(X)$.

1.

The first step is to examine the data set `physical1.csv`, to see if the two processes are related to each other.

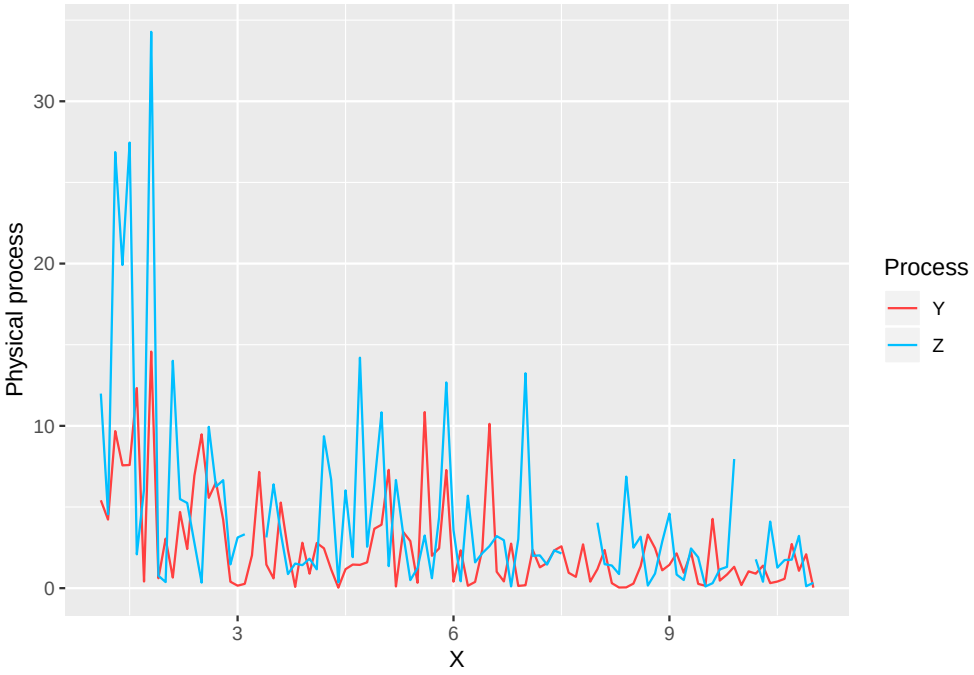

Figure 3: Time series plot of the dependence of Z and Y versus X.

In Figure 3 it seems that the two processes are related to each other, with respect to X, since the graphs follows similar patterns. We can also observe that the physical process Z has a greater variation, especially at the beginning of the series, but also in general, compared to the process Y.

2.

Using the following model,

$$Y_i \sim exp\left(\frac{X_i}{\lambda}\right)$$
$$Z_i \sim exp\left(\frac{X_i}{2\lambda}\right)$$

where λ is an unknown parameter, we derive the EM algorithm to estimate λ.

Y and Z are defined by the following density functions,

$$\begin{aligned} Z(X) &= \frac{X_i}{2\lambda} exp\left(-\frac{X_i}{2\lambda} Z_i\right) \\ Y(X) &= \frac{X_i}{\lambda} exp\left(-\frac{X_i}{\lambda} Y_i\right) \end{aligned}$$

Assuming that $Y(X)$ and $Z(X)$ are i.i.d. we can obtain the joint density function,

$$\begin{aligned} f(Y,Z) &= \frac{X_i}{2\lambda} exp\left(-\frac{X_i}{2\lambda}Z_i\right) * \frac{X_i}{\lambda} exp\left(-\frac{X_i}{\lambda}Y_i\right) \\ &= \frac{X_i^2}{2\lambda^2} exp\left[-\frac{X_i(Z_i+2Y_i)}{2\lambda}\right] \end{aligned}$$

From this point, we compute the likelihood of the distribution,

$$\mathcal{L}(Y,Z|X,\lambda) = \prod_{i=1}^{n} \frac{X_i^2}{2\lambda^2} exp\left[-\frac{X_i(Z_i+2Y_i)}{2\lambda}\right]$$

It is easier to work with the logarithm of the likelihood, therefore,

$$\begin{aligned} log\mathcal{L}(Y,Z|X,\lambda) &= log\prod_{i=1}^{n} \frac{X_i^2}{2\lambda^2} exp\left[-\frac{X_i(Z_i+2Y_i)}{2\lambda}\right] \\ &= -nlog(2\lambda^2) + \sum_{i}^{n} log(X_i^2) - \sum_{i}^{n} \frac{X_i}{2\lambda}(Z_i+2Y_i) \end{aligned}$$

E-Step

$$\begin{aligned} Q(\lambda, lambda^k) &= E\left[log\mathcal{L}(Y,Z|X,\lambda)\right] \\ &= -nlog(2\lambda^2) + \sum_{1}^{n} log(X_i^2) - \frac{1}{2\lambda}\sum_{1}^{r} X_i(Z_i+2Y_i) - \frac{1}{2\lambda}\sum_{r+1}^{n} X_i(E[Z_i]+2Y_i) \\ &= -nlog(2\lambda^2) + \sum_{1}^{n} log(X_i^2) - \frac{1}{2\lambda}\sum_{1}^{r} X_i(Z_i+2Y_i) - \frac{1}{2\lambda}\sum_{r+1}^{n} X_i(\frac{2\lambda^k}{X_i}+2Y_i) \\ &= -nlog(2\lambda^2) + \sum_{1}^{n} log(X_i^2) - \frac{1}{2\lambda}\sum_{1}^{r} X_i(Z_i+2Y_i) - \frac{1}{\lambda}\sum_{r+1}^{n} (\lambda^k + X_iY_i) \end{aligned}$$

M-Step

$$\begin{aligned} \frac{\partial Q(\lambda,\lambda^k)}{\partial\lambda} &= 0 \\ \frac{\partial Q(\lambda,\lambda^k)}{\partial\lambda} &= -\frac{2n}{\lambda} + \frac{1}{2\lambda^2}\sum_{1}^{r} X_i(Z_i+2Y_i) + \frac{1}{\lambda^2}\sum_{r+1}^{n} (\lambda^k + X_iY_i) \end{aligned}$$

Therefore,

$$\lambda = \frac{\sum_1^r X_i(Z_i+2Y_i) + 2\sum_{r+1}^n (\lambda^k + X_iY_i)}{4n}$$

The code can be found in the Appendix.

3.

Once we have the EM algorithm, we run it with initial value $\lambda_0 = 100$ and convergence criterion "stop if the change in λ is less than 0.001". This yields the following results.

```
## number_of_iterations       optimal_lambda
##              5.00000             10.69566
```

As we can see in only 5 iterations we reached the optimal λ.

4.

As a final task, we are going to see if the optimal value $\lambda = 10.69566$ is reasonable for the physical processes $Y = Y(X)$ and $Z = Z(X)$. We examine the results by computing the expected values of Y and Z, which is given by $E[Y_i] = \frac{\lambda}{X_i}$ and $E[Z_i] = \frac{2\lambda}{X_i}$, respectively.

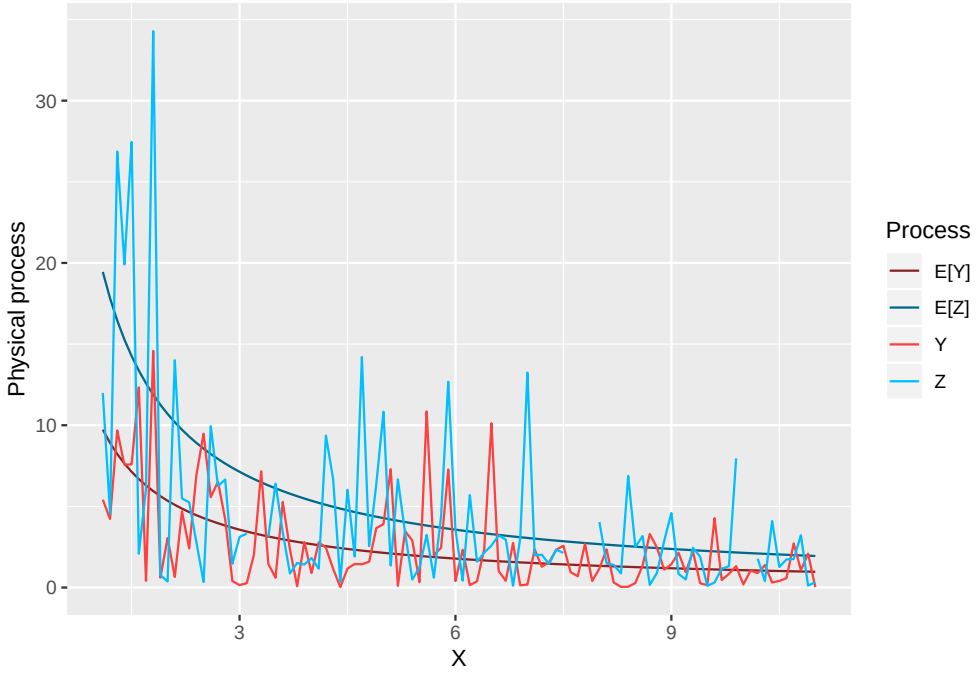

Figure 4: Time series plot of E[Y] and E[Z] versus X, and the dependence of Z and Y versus X.

In Figure 4, we can see that $E[Y]$ and $E[Z]$ versus X are in the same plot as Y and Z versus X. From this plot it is clear that the optimal λ is reasonable, since the exponential curves $E[Y]$ and $E[Z]$, the follow their corresponding values of Y and Z.

Appendix

```
knitr::opts_chunk$set(echo = FALSE)
# R version
RNGversion('3.5.1')
library("ggplot2")
#1.1
f <- function(x){
  return(x^2/exp(x) - 2*exp(-1*(9*sin(x)) / (x^2 + x + 1)))
}

#1.2
crossover <- function(x,y){
  return((x+y) / 2)
}

#1.3
mutate <- function(x){
  return(x^2 %% 30)
}
#1.4
genetic <- function(maxiter, mutprob){
  #a
  plot(x = seq(0,30, by=0.1), y = f(seq(0,30, by=0.1)),
       type = "l", xlab = "x", ylab = "f(x)",
       main = paste("maxiter:",maxiter,"mutprob:",mutprob))
  abline(v=seq(0,30, by = 0.1)[which.max(f(seq(0,30,by = 0.1)))], col="blue" )
  #maximum value: f(1.2) = 0.234

  #b
  X = seq(0,30,5)

  #c
  Values = f(X)

  #d
    #set seed
    set.seed(1234567890)
    max=max(Values)
    for (i in 1:maxiter) {
      #i
      parents = match(sample(X, 2),X)

      #ii
      victim = order(Values)[1]

      #iii
      kid = crossover(X[parents[1]],X[parents[2]])
      p = runif(1)
      if (p < mutprob) {
        kid = mutate(kid)
      }
```

```
      #iv
      X[victim] = kid
      Values = f(X)

      #v
      max = max(Values)
      }

  #e
  points(x = X, y = Values, col = "red", pch = 19)
  return(max)
}

#1.5
plot(x = seq(0,30, by=0.1), y = f(seq(0,30, by=0.1)),
       type = "l", xlab = "x", ylab = "f(x)")
abline(v=seq(0,30, by = 0.1)[which.max(f(seq(0,30,by = 0.1)))], col="blue" )
points(x = seq(0,30,5), y = f(seq(0,30,5)), col = "red", pch = 19)


max_values = data.frame(matrix(nrow = 3, ncol = 2))
par(mfrow=c(3,2))
max_values[1,1] = genetic(10,0.1)
max_values[1,2] = genetic(100,0.1)
max_values[2,1] = genetic(10,0.5)
max_values[2,2] = genetic(100,0.5)
max_values[3,1] = genetic(10,0.9)
max_values[3,2] = genetic(100,0.9)

rownames(max_values) = c("mutprob 0.1","mutprob 0.5","mutprob 0.9")
colnames(max_values) = c("maxiter 10","maxiter 100")
knitr::kable(max_values, caption = 'Maximum values')

#QUESTION 2
# 2.1
physical <- read.csv2("physical1.csv", sep = ",")

X <- as.numeric(as.character(physical$X))
Y <- as.numeric(as.character(physical$Y))
Z <- as.numeric(as.character(physical$Z))

data <- data.frame(X = c(X,X), value = c(Y,Z), Process= rep(c("Y","Z"), each= 100))

# var(Z,na.rm = TRUE)
# var(Y,na.rm = TRUE)

# Time series plot
ggplot(data = data, aes(x = X, y = value, col = Process)) +
  geom_line() +
  ylab("Physical process")+
  scale_color_manual(values=c( "brown1","deepskyblue1"))

## 2.2
```

```r
EM_algorithm <- function(X, Y, Z, lambda, eps, k_max) {
    Z_obs <- Z[!is.na(Z)]
    Z_miss <- Z[is.na(Z)]

    X_obs <- X[!is.na(Z)]
    X_miss <- X[is.na(Z)]

    Y_obs <- Y[!is.na(Z)]
    Y_miss <- Y[is.na(Z)]

    n <- length(c(Z_obs, Z_miss))
    r <- length(Z_obs)

    k<-1
    lambda_prev <- lambda+10+100*eps #random number to initialize the algorithm
    lambda_curr <- lambda

    while (k<k_max+1 && abs(lambda_prev-lambda_curr)>=eps) {

        lambda_prev<-lambda_curr

        ## E-step
        EY <- -n*log(2*lambda_prev^2) + sum(log(X^2)) -
          sum(X_obs*(Z_obs+2*Y_obs))/(2*lambda_prev) -
          sum(lambda_curr+X_miss*Y_miss)/lambda_prev

        ## M-step
        lambda_curr <- (sum(X_obs*(Z_obs+2*Y_obs)) + 2*sum(lambda_curr+X_miss*Y_miss))/(4*n)

        k<-k+1

    }
    return(c(number_of_iterations = k-1, optimal_lambda = lambda_curr))
}

##2.3

EM_algorithm(X,Y,Z, 100, 0.001, 100)
# 2.4
optimal_lambda <- EM_algorithm(X,Y,Z, 100, 0.001, 100)[2]
# Expected values
EY <- optimal_lambda/X
EZ <- 2*optimal_lambda/X

data2 <- data.frame(X = c(X,X), value = c(Y,Z,EY,EZ), Process= rep(c("Y","Z","E[Y]","E[Z]"),
                                                                   each= 100))

#
ggplot(data = data2, aes(x = X, y = value, col = Process)) +
  geom_line() +
  ylab("Physical process") +
  scale_color_manual(values=c("brown4", "deepskyblue4", "brown1", "deepskyblue1"))
```