Computational Statistics (732A90) Lab06

Christoforos Spyretos, Marc Braun, Marketos Damigos, Patrick Siegfried Hiemsch & Prakhar

2021-12-07

Question 1: Genetic algorithm

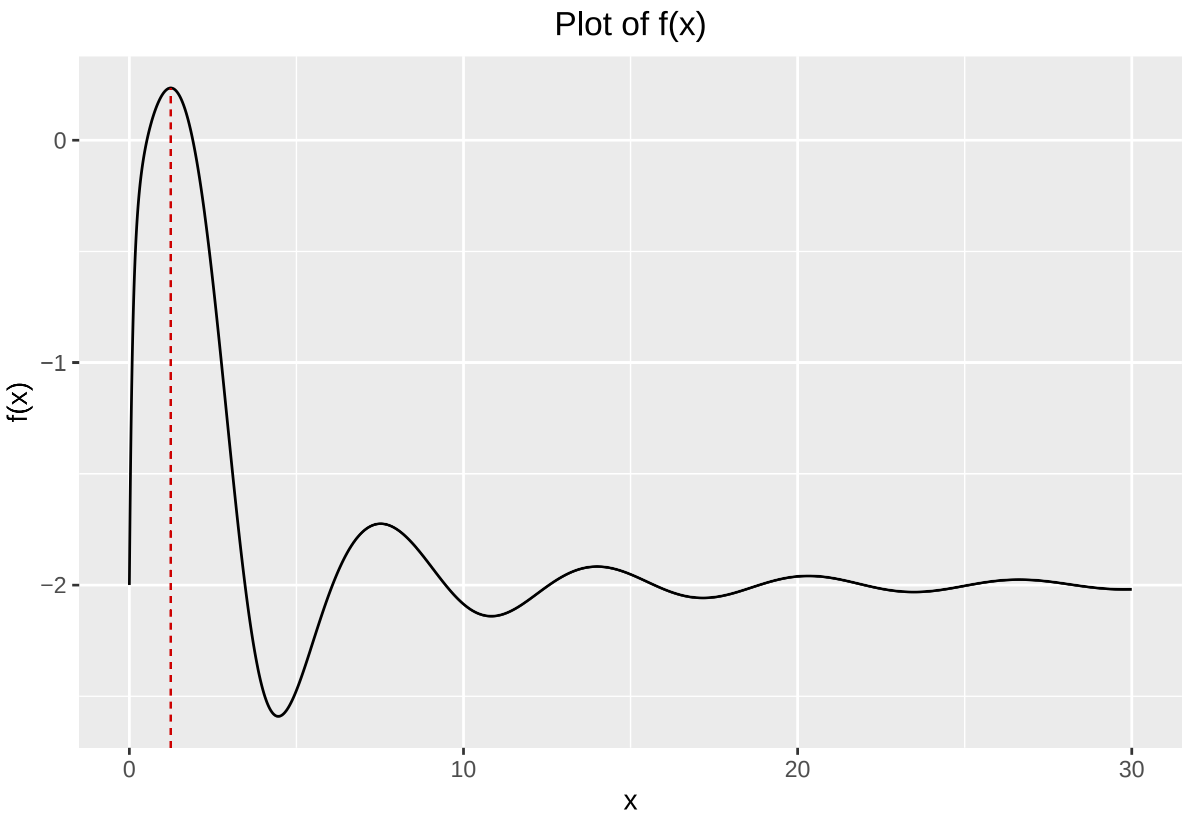
Task 1

In this exercise we want to perform one-dimensional maximization with the help of a genetic algorithm. The function we want to optimize is $f(x)$:

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

The interval, in which we will search for the optimum is $[0, 30]$. To get a better overview, we plot the function $f(x)$ over this interval.

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

library(ggplot2)

max <- optim(my_f, par = 0, lower = 0, upper = 30, control = list(fnscale = -1),
    method = "L-BFGS-B")
max_x <- max$par
max_y <- max$value

x <- seq(0, 30, 0.01)

f_plot <- ggplot(data = data.frame(x = x, y = my_f(x))) + geom_line(aes(x, y)) +
    geom_segment(aes(x = max_x, y = -Inf, xend = max_x, yend = max_y), color = "red3",
        linetype = "dashed", size = 0.3) + ggtitle("Plot of f(x)") + ylab("f(x)") +
    theme(plot.title = element_text(hjust = 0.5))

f_plot
```



Just from a visual analysis, it is obvious that the function reaches the maximum value in the interval between 0 and 5. The exact value found by optimizing $f(x)$ with the optim-function is 1.2391562 marked by the red dashed line in the plot.

Task 2

To prepare our genetic algorithm, we first implement the two functions to perform crossovers and mutations and then create a seperate function, that depends on the parameters maxiter (number of iterations) and mutprob (probability of mutation in an iteration) and executes the genetic maximization.

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

Task 3

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

Task 4

```
my_f4 = function(maxiter, mutprob) {

    # Part a)
    plot = ggplot() + geom_function(fun = "my_f") + xlim(0, 30)
```

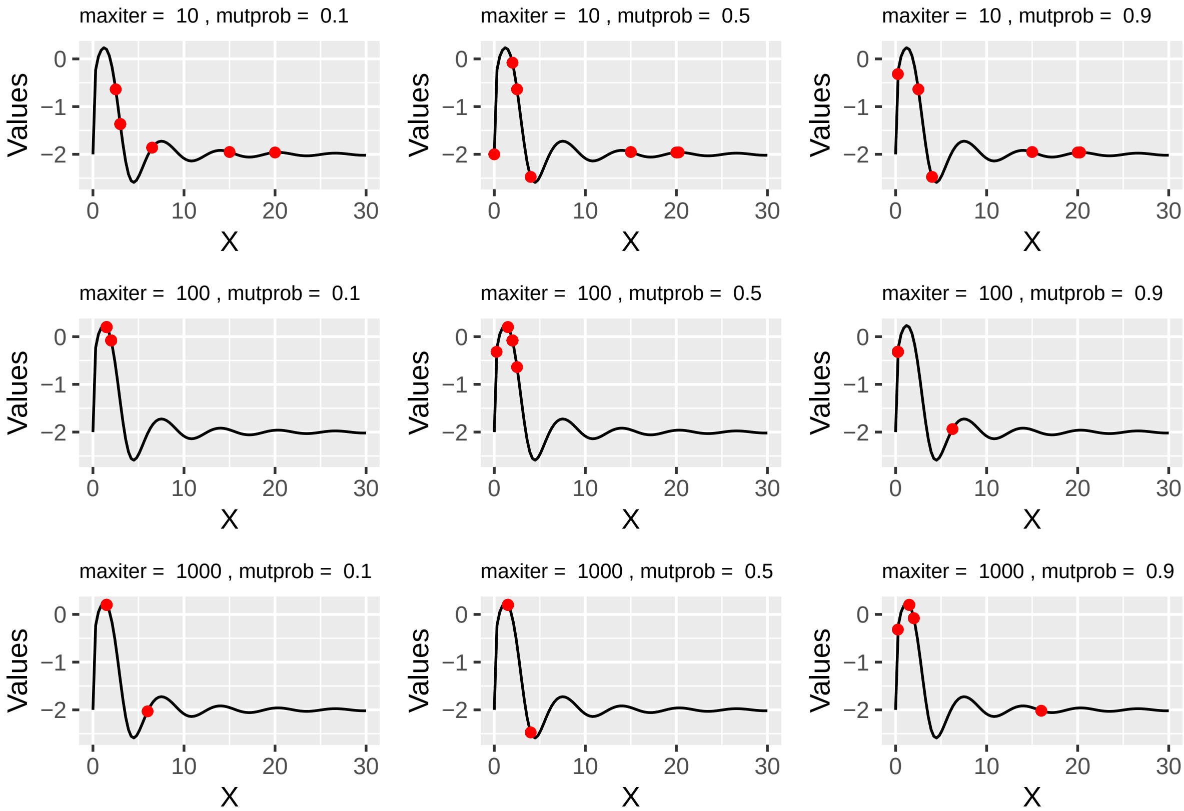
```r
    # Part b)
    X = seq(0, 30, 5)

    # Part c)
    Values = my_f(X)

    # Part d)
    max_value = NA
    for (i in 1:maxiter) {
        # i)
        parents = sample(1:length(X), 2)

        # ii)
        victim = which.min(Values)

        # iii)
        kid = crossover(parents[1], parents[2])

        if (mutprob > runif(1, 0, 1)) {
            kid = mutate(kid)
        }

        # iv)
        X[victim] = kid

        Values = my_f(X)

        # v)
        max_value = max(Values)
    }

    plot = plot + geom_point(data = data.frame(X = X, Values = Values), aes(X, Values),
        color = "red")

    return(plot)
}
```

Task 5

To test the implemented algorithm we test it with different values for the two parameters and plot the results in a grid to get a good overview of the optimization outcomes.

```r
library(gridExtra)

maxiter_params <- c(10, 100, 1000)
mutprob_params <- c(0.1, 0.5, 0.9)
plot_list = list()
k_plot <- 1
set.seed(12345)
for (i in maxiter_params) {
    for (j in mutprob_params) {
        plot <- my_f4(maxiter = i, mutprob = j) + ggtitle(paste("maxiter = ", i,
            ", mutprob = ", j)) + theme(plot.title = element_text(size = 8))

        plot_list[[k_plot]] <- plot
        k_plot = k_plot + 1
```

```
    }
}

grid.arrange(grobs = plot_list, nrow = 3, ncol = 3)
```

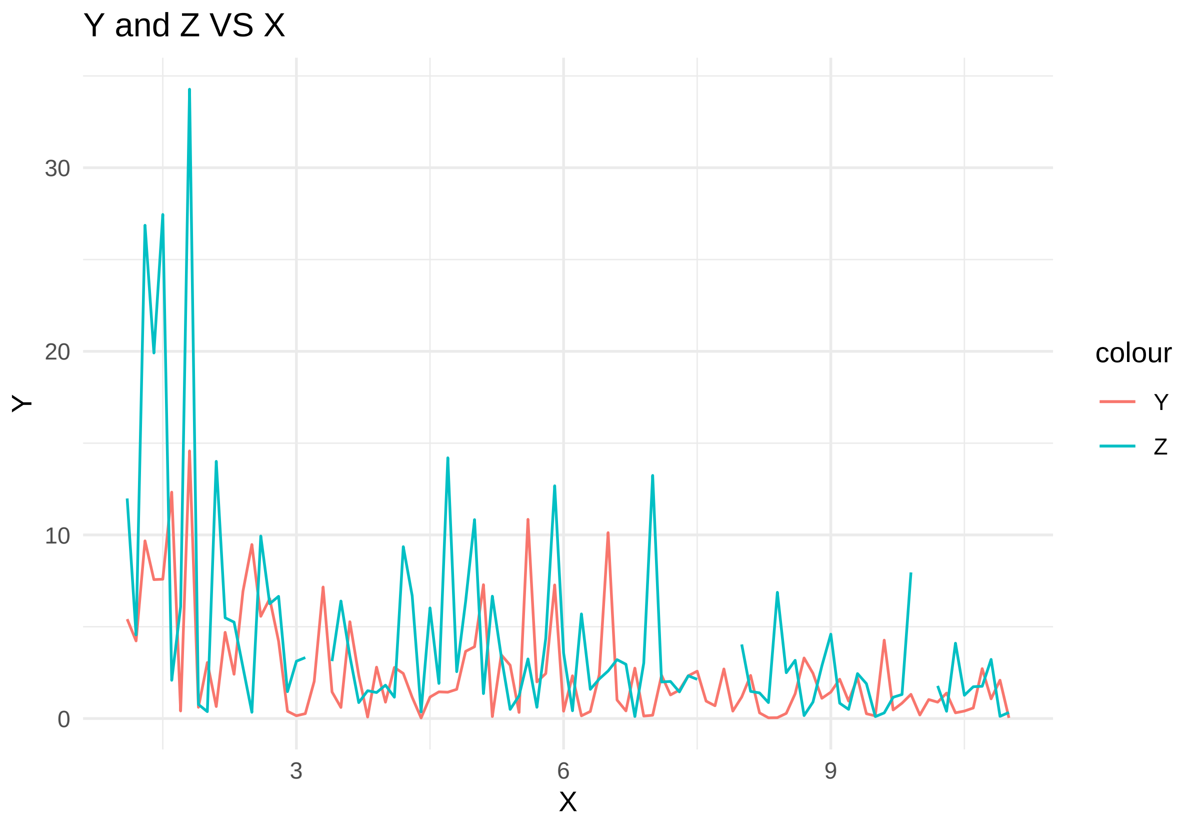


From the above plot we can see, that not all runs of the algorithm found the global optimum (in relation to the interval $[0, 30]$). If a small number of iterations is chosen, the population points marked in red are still very widely distributed. But with increasing iterations, we get better results in general (even with a very low mutation rate of 0.1 the algorithm finds a point close to the global optimum). For the probability of mutation, we can observe that extreme values (0.1/0.9) show worse results than 0.5, where after 100 iterations and also after 1000 iterations the global optimum point is found by the algorithm (all population points have very similar x-value which are close to the optimal x we computed earlier).

Question 2: EM algorithm

Task 1

```
physical = read.csv("physical1.csv")

ggplot(data = physical) + geom_line(aes(x = X, y = Y, color = "Y")) + geom_line(aes(x = X,
    y = Z, color = "Z")) + ggtitle("Y and Z VS X") + theme_minimal()
```



The two variables seem to be related but with Z to have spikes of higher magnitude. We can notice that with increasing X we get smaller values for both Y and Z, especially in the area after 6 we get much less variation for the two variables. Also we can mention some missing values for Z.

Task 2

```
my_lambda_est = function(data, l_0, conv) {
    l_prev = l_0
    n = nrow(data)
    u = which(is.na(data$Z))
    l_cur = (sum(data$X * data$Y) + (0.5 * sum(data$X[-u] * data$Z[-u])) + (length(u) *
        l_prev))/(2 * n)
    counter = 1

    while (abs(l_prev - l_cur) >= conv) {
        l_prev = l_cur
        l_cur = (sum(data$X * data$Y) + (0.5 * sum(data$X[-u] * data$Z[-u])) + (length(u) *
            l_prev))/(2 * n)
```

```
        counter = counter + 1
    }

    res = list(opt_l = l_cur, itterations = counter)
    return(res)
}
```

```
my_lambda_est(physical, 100, 0.001)
```

```
## $opt_l
## [1] 10.69566
##
## $itterations
## [1] 5
```

The optimal $\lambda = 10.69566$ and we needed 5 iterations to calculate it.