Computational Statistics (732A90) Lab06

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

2021-12-03

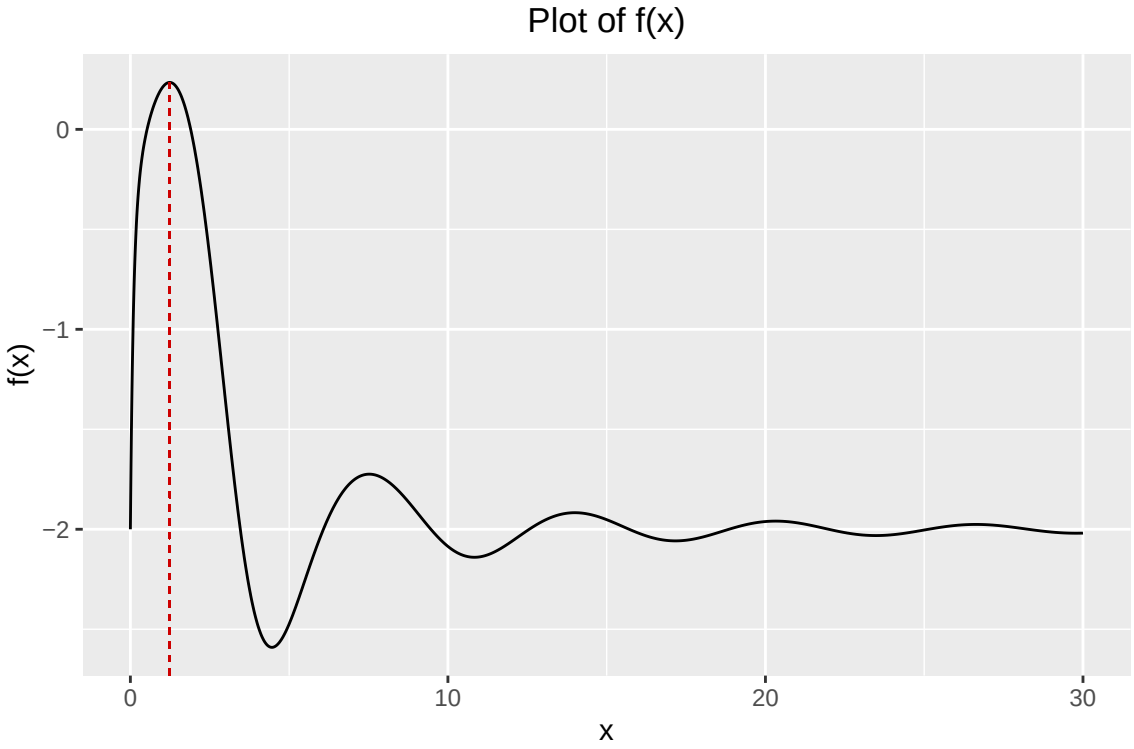
Question 1

Part 1 - Exploring the function

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

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

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



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.

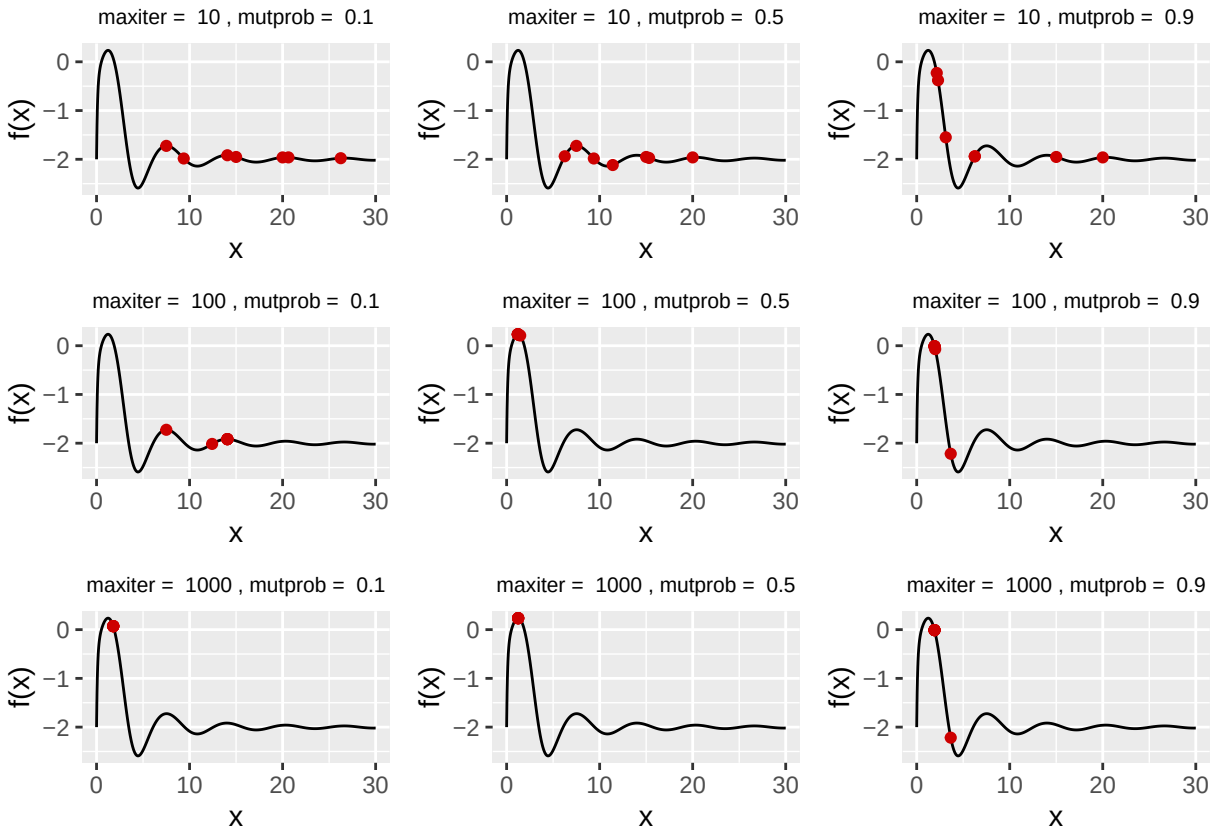
Part 2 - Genetic Algorithm

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)
}

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

```
genetic_optimizer <- function(maxiter, mutprob) {

    f_plot <- ggplot(data = data.frame(x = x, y = f(x))) + geom_line(aes(x,
        y)) + ylab("f(x)") + theme(plot.title = element_text(hjust = 0.5))

    X <- seq(0, 30, 5)
    Values <- f(X)

    max_obj_value <- 0

    set.seed(12345)

    for (i in 1:maxiter) {
        parents <- sample(X, 2, length(X))
        victim_index <- order(Values)[1]
        kid <- crossover(parents[1], parents[2])

        mutate <- runif(1)

        if (mutate <= mutprob) {
            kid <- mutate(kid)
        }

        X[victim_index] <- kid
        Values <- f(X)
        max_obj_value <- max(Values)
    }

    f_plot <- f_plot + geom_point(data = data.frame(X = X, Values = Values),
        aes(X, Values), color = "red3")

    return(f_plot)
}
```

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.



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 then for 0.5, where after 100 iterations and also after 1000 iterations the global optimum point is found by the algorithm (all population points have the same x-value which equals the optimal x we computed earlier).