CompStatLab2

Xuan Wang & Priyarani Patil

2023-11-07

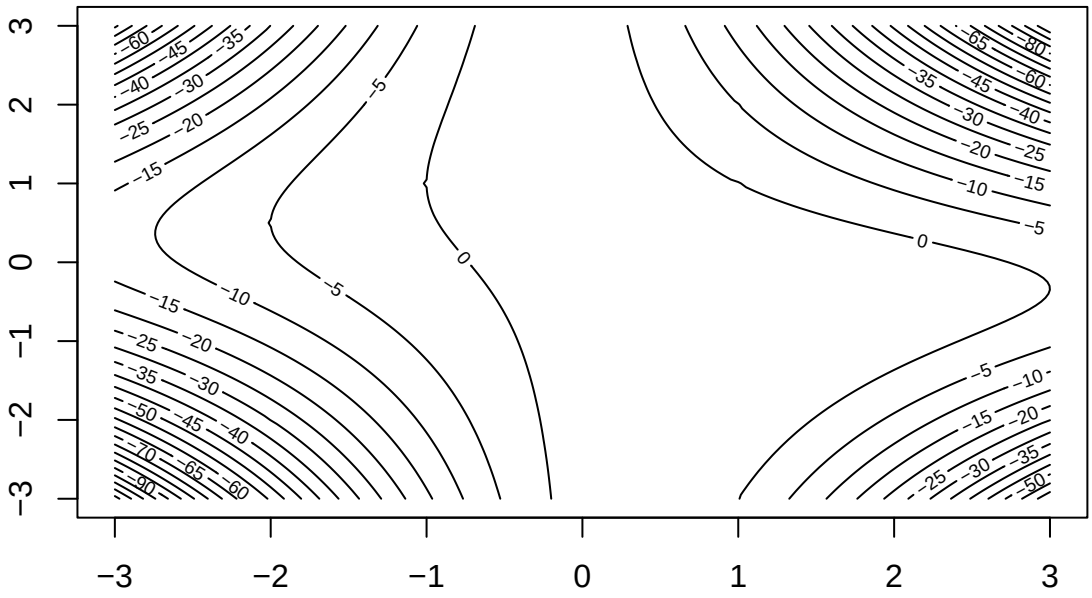
Computational Statistics 732A90 | Computer Lab 1

Question 1: Optimisation of a two-dimensional function

Consider the function $g(x, y) = -x^2 - x^2y^2 - 2xy + 2x + 2$ *It is desired to determine the point (x, y), x, y [−3, 3], where the function is maximized.*

a. Derive the gradient and the Hessian matrix in dependence of x, y. Produce a contour plot of the function g.

Answer:



b. Write an own algorithm based on the Newton method in order to find a local maximum of g

Answer:

c. Use different starting values: use the three points (x, y) = (2, 0), (−1, −2), (0, 1) and a fourth point of your choice. Describe what happens when you run your algorithm for each of those starting values. If your algorithm converges to points (x, y), compute the gradient and the Hessian matrix at these points and decide about local maximum, minimum, saddle point, or neither of it. Did you find a global maximum for x, y [−3, 3]?

Answer:

d. What would be the advantages and disadvantages when you would run a steepest ascent algorithm instead of the Newton algorithm?

Answer:

Question 2

a. Write a function for an ML-estimator for (0, 1) using the steepest ascent method with a step-size reducing line search (back-tracking). For this, you can use and modify the code for the steepest ascent example from the lecture. The function should count the number of function and gradient evaluations.

Answer:

b. Compute the ML-estimator with the function from a. for the data (xi, yi) above. Use a stopping criterion such that you can trust five digits of both parameter estimates for 0 and 1. Use the starting value (0, 1) = (−0.2, 1). The exact way to use backtracking can be varied. Try two variants and compare number of function and gradient evaluation done until convergence.

Answer:

c. Use now the function optim with both the BFGS and the Nelder-Mead algorithm. Do you obtain the same results compared with b.? Is there any difference in the precision of the result? Compare the number of function and gradient evaluations which are given in the standard output of optim.

Answer:

d. Use the function glm in R to obtain an ML-solution and compare it with your results before.

Answer:

Appendix:

two_dimensional.R

```r
# function g to be maximized
g <- function(x,y) {
  return(-x^2 - x^2 * y^2 - 2 * x * y + 2 * x + 2)
}

# partial derivative x for function g
deriv_x <- function(x, y) {
  return(-2 * x - 2 * x * y^2 - 2 * y + 2)
}

# partial derivative y for function g
deriv_y <- function(x, y) {
  return(-2 * x^2 * y - 2 * x)
}

# gradient for function g
gradient <- function (x, y) {
  return(c(deriv_x(x,y), deriv_y(x,y)))
}

# second partial derivative x for function g
deriv_x_11 <- function(x, y) {
  return(-2 - 2 * y^2)
}

# second partial derivative x/y for function g
```

```r
deriv_x_12 <- function(x, y) {
  return(-4 * x * y - 2)
}

# second partial derivative y for function g
deriv_y_22 <- function(x, y) {
  return(-2 * x^2)
}

# hessian matrix for function g
hessian_g <- function(x, y) {
  matrix(c(deriv_x_11(x,y), deriv_x_12(x,y), deriv_x_12(x,y), deriv_y_22(x, y), nrow=2, ncol=2))
}

# produce a contour plot
xgrid <- seq(-3,3, by=0.05)
ygrid <- seq(-3,3, by=0.05)
length_x <- length(xgrid)
length_y <- length(ygrid)
dxy <- length_x * length_y
gxy <- matrix(rep(NA,dxy), nrow = length_x)
for ( i in 1:length_x) {
  for ( j in 1:length_y) {
    gxy[i, j] <- g(xgrid[i], ygrid[j])
  }
}

mgxy <- matrix(gxy, nrow = length_x, ncol = length_y)
contour(xgrid, ygrid, mgxy, nlevels=40)

newton_g <- function(x,y) {

}

g <- function(x,y) {

}
```