

Lab 4

Gabriel J. Young

June 5th, 2017

Instructions

Make sure that you upload an RMarkdown file to the canvas page (this should have a .Rmd extension) as well as the PDF output after you have knitted the file (this will have a .pdf extension). Note that since you have already knitted this file, you should see both a **Lab4_UNI.pdf** and a **Lab4_UNI.Rmd** file in your UN4206 folder. Click on the **Files** tab to the right to see this. The files you upload to the Canvas page should be updated with commands you provide to answer each of the questions below. You can edit this file directly to produce your final solutions. The lab is due 11:59pm on Friday, June 9th.

Optimization

The goal of this lab is to write a simple optimization function in **R** which estimates the global minimum of a convex differentiable function $f(x)$. Specifically, consider the function

$$f(x) = \frac{-\log(x)}{1+x}, \quad x > 0,$$

where $\log(x)$ is the natural logarithm of x . We seek to estimate the value of $x > 0$ such that $f(x)$ achieves its global minimum. For example, the global minimum of the function $g(x) = x^2 - 4x + 3$ is at $x = 2$. The minimum of $g(x)$ can easily be computed using the vertex formula for quadratic functions, i.e., $x = -b/(2a) = 4/(2 * 1) = 2$. In most cases, the minimum does not have a closed form solution and must be computed numerically. Hence we seek to estimate the global minimum of $f(x)$ numerically via gradient descent.

Tasks

- 1) Using **R**, define the function

$$f(x) = \frac{-\log(x)}{1+x}, \quad x > 0.$$

Test the points $f(0)$ and $f(2)$.

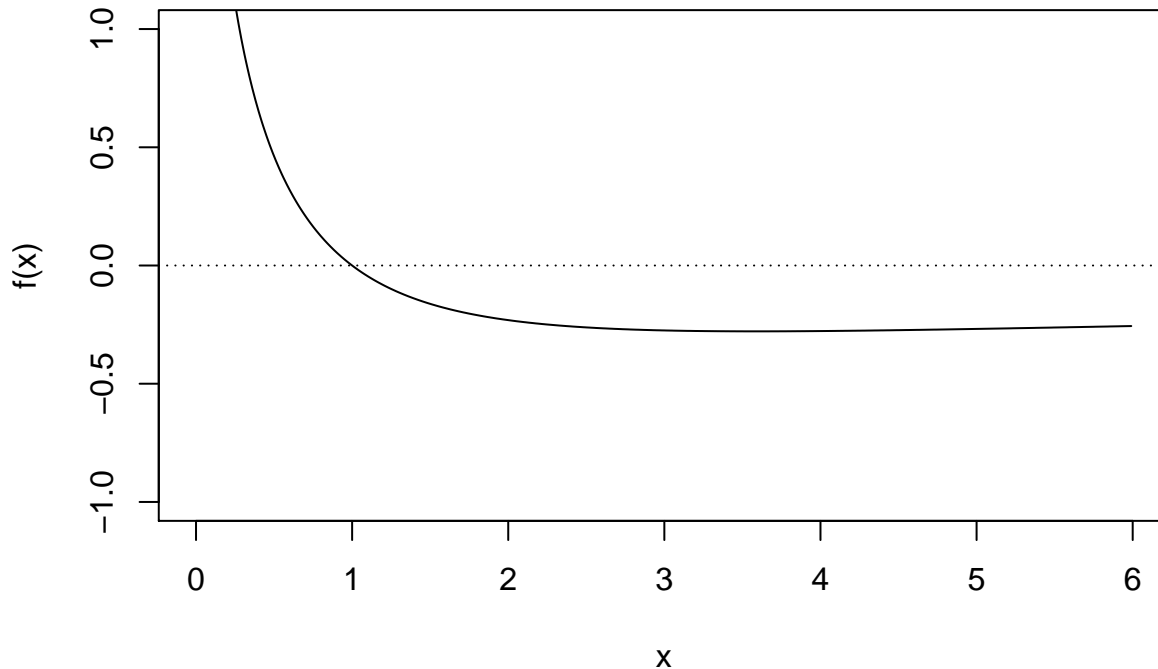
```
f <- function(x) {  
  return(-log(x)/(1+x))  
}  
f(0)
```

```
## [1] Inf
```

```
f(2)
```

```
## [1] -0.2310491
```

2) Plot the function $f(x)$ over the interval $(0, 6]$.



3) By inspection, where do you think global minimum is located at?

Somewhere around 3.5

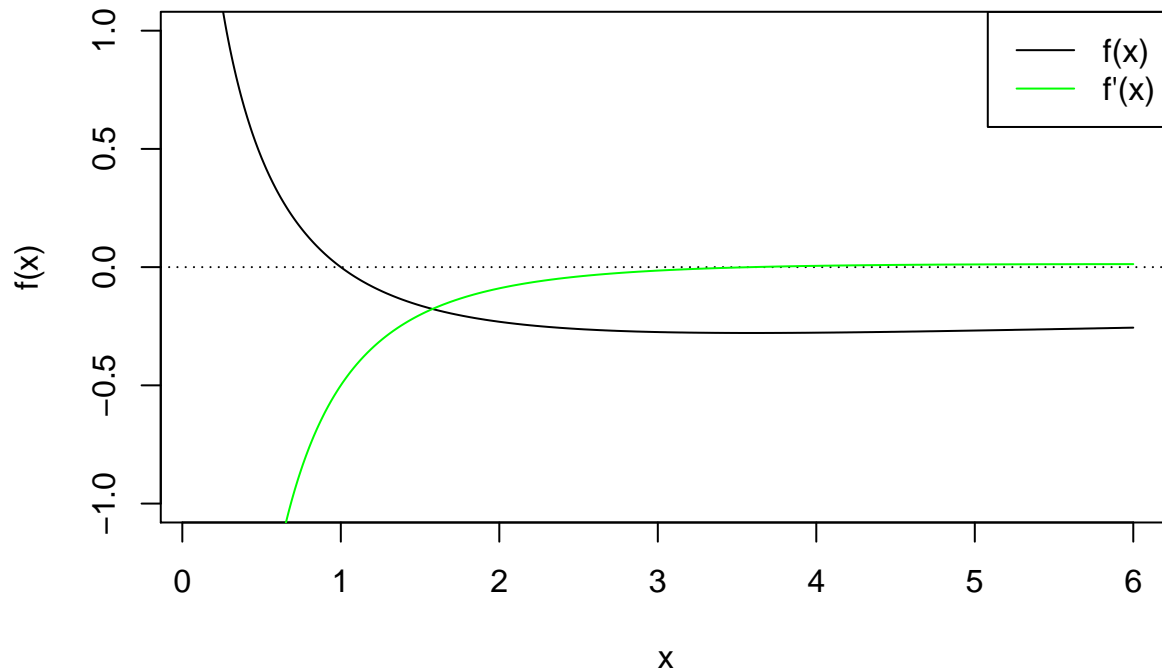
4) Define a **R** function which computes the difference quotient of $f(x)$, i.e., for $h > 0$,

$$\frac{f(x+h) - f(x)}{h}.$$

This function should have two inputs; h and x . Name the difference quotient function **diff.quot**. Note that for small h , this function is the approximate derivative of $f(x)$.

```
diff.quot <- function(x,h=.0001) {  
  return((f(x+h)-f(x))/h)  
}
```

```
x <- seq(from=.10,to=6,.01)  
plot(x,f(x),type="l",ylim=c(-1,1))  
abline(h=0,lty=3)  
lines(x,diff.quot(x),col="green")  
legend("topright",legend=c("f(x)", "f'(x)"),lty=c(1,1),col=c("black", "green"))
```



- 5) Plot both the difference quotient function **diff.quot** and $f(x)$ over the interval $(0, 6]$. Fix $h = .0001$ to construct this plot. Comment on any interesting features.
- 6) Write a **R** function named **basic.grad.descent** that runs the basic gradient descent algorithm on the function $f(x)$. The function should have inputs:
 - (a) Initial value **x**
 - (b) Maximum iterations **max.iter** with default 10000.
 - (c) Stopping criterion **stop.deriv** with default $1e-10$.
 - (d) Derivative step size **h** with default $.0001$.
 - (e) Step size **step.scale** with default $.5$.

The function should have outputs:

- (a) The value x that yields the minimum of $f(x)$.
- (b) The minimum value of $f(x)$.
- (c) The number of iterations the algorithm took to reach the minimum.
- (d) A logical indicator displaying whether or not the algorithm converged.

Hints

- I) The main idea of this algorithm is to update x using the relation

$$x_{i+1} = x_i - \text{step.scale} * f'(x_i),$$

where $f'(x_i)$ is approximated by the difference quotient.

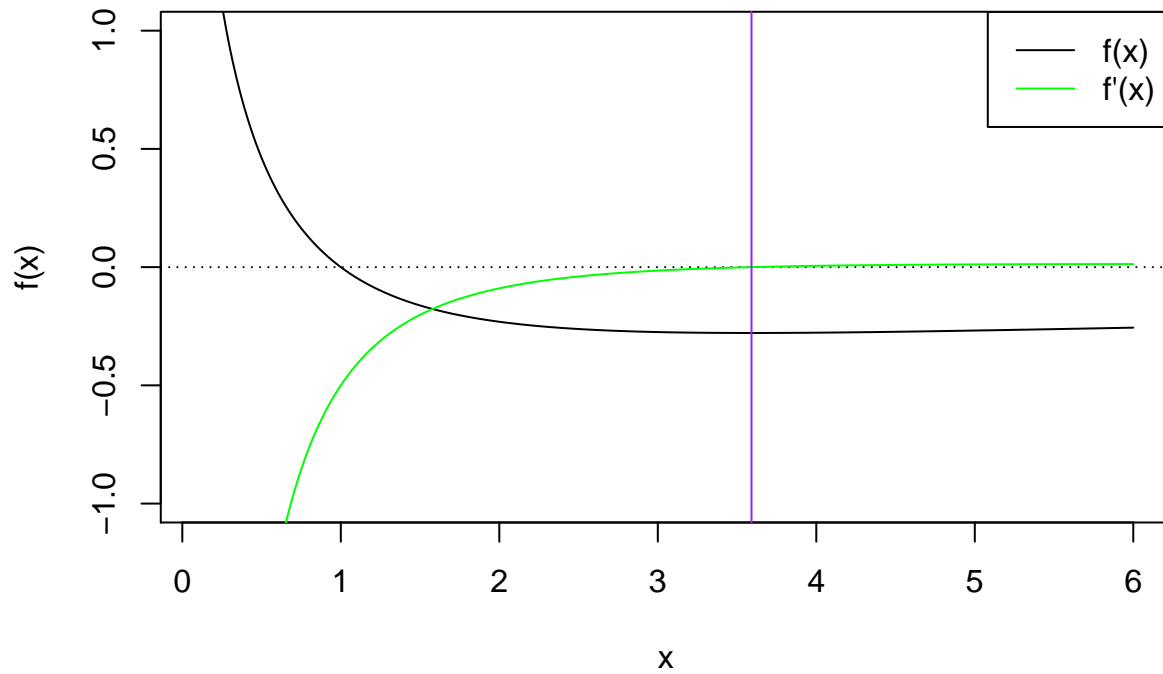
- II) Build your function using the sample code from slide 51 in lecture notes **UN2102Day17**.

Note: This lab is expected to take longer than the scheduled in-class contact hours.

```
uni.grad.decent <- function(x,
                             max.iter = 10000,
                             stop.deriv = 1e-10,
                             h = .0001,
                             step.scale = .5) {
  iter <- 0
  deriv <- Inf
  for (i in 1:max.iter) {
    iter <- iter + 1
    deriv <- (f(x + h) - f(x))/h
    x <- x - step.scale*deriv
    if (abs(deriv) < stop.deriv) {break()}
  }
  fit <- list(x = x,
              iteration = iter,
              converged = (iter < max.iter))
  return(fit)
}
uni.grad.decent(x=1)
```

```
## $x
## [1] 3.591071
##
## $iteration
## [1] 2201
##
## $converged
## [1] TRUE
```

```
x <- seq(from=.10,to=6,.01)
plot(x,f(x),type="l",ylim=c(-1,1))
abline(h=0,lty=3)
lines(x,diff.quot(x),col="green")
legend("topright",legend=c("f(x)", "f'(x)"),lty=c(1,1),col=c("black","green"))
abline(v=uni.grad.decent(x=1)$x,col="purple")
```



7) Check the optimal value using the base **R** function `nlm()`.

```
nlm(f,p=1)
```

```
## $minimum
## [1] -0.2784645
##
## $estimate
## [1] 3.591119
##
## $gradient
## [1] -1.273731e-08
##
## $code
## [1] 1
##
## $iterations
## [1] 10
```