Gov 2018 Problem Set 1: Gradient Descent

Alexandra Norris

Tuesday February 8, 2022

Please upload answers to all questions (including computational questions and any related graphics and R code *with comments*, in .rmd *and* knitted pdf forms to Gradescope by Tuesday before class. For any problems that require calculations, please show your work. Finally, when doing simulations or draws from a random distribution in `R`, please set your seed immediately using `set.seed(2022)`.

Gradient Descent

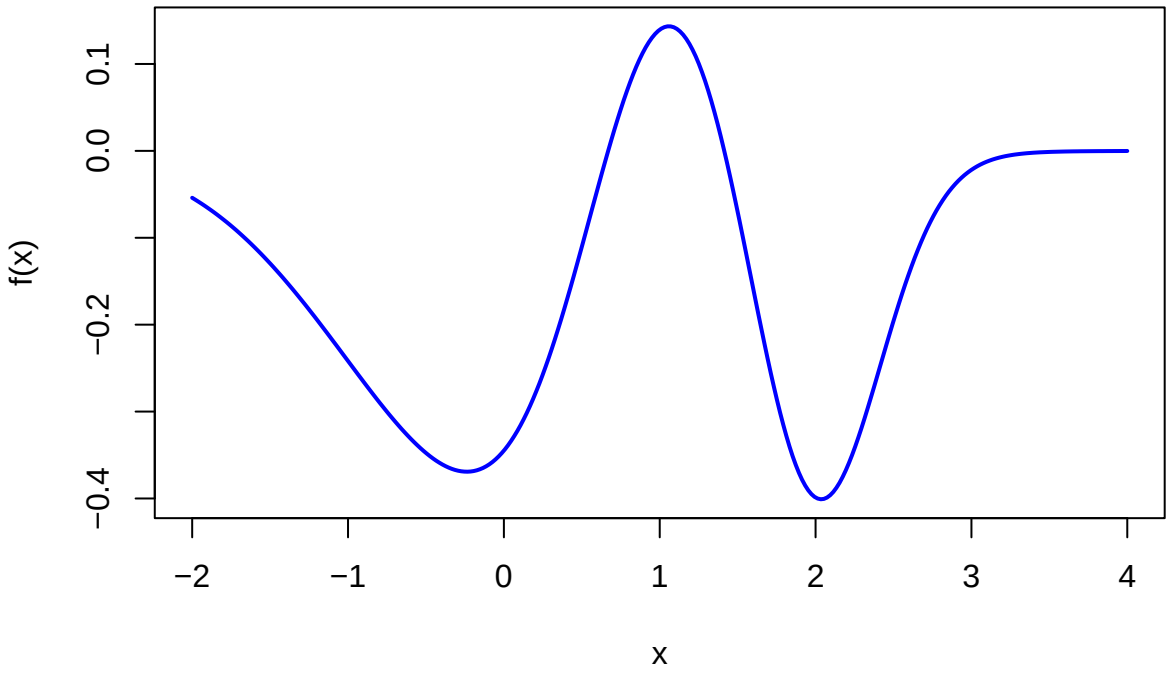
In this problem, we use gradient descent to approximate the minima of the function,

$$f(x) = -\frac{1}{\sqrt{2\pi}}\exp\big(-\frac{1}{2}x^2\big) + \frac{1}{\sqrt{2\pi}}\exp\big(-\frac{1}{2}(\frac{x-1}{0.5})^2\big) - \frac{1}{\sqrt{2\pi}}\exp\big(-\frac{1}{2}(\frac{x-2}{0.4})^2\big)$$

Question 1 [1 pt]

Plot the function $f(x)$ for $x \in (-2, 4)$.

```
# make the equation a function
f <- function(x){
  y = -(1/sqrt(2*pi))*exp(-(1/2)*x^2) + (1/sqrt(2*pi))*exp((-1/2)*(((x-1)/0.5)^2)) - (1/sqrt(2*pi))*exp
  return(y)
}

# plot the curve using the curve function and look at x values between -2 and 4
curve(f(x), from=-2, to=4, n=300, xlab="x", ylab="f(x)", col="blue", lwd=2, main="Plot of Equation")
```



Question 2 [2 pts]

We will approximate the derivative $f'(x)$ by replacing f by a linear function within a small window. We choose some small value $\delta > 0$ and compute:

$$\hat{f}_\delta(x) := \frac{f(x+\delta) - f(x-\delta)}{|[x-\delta, x+\delta]|} = \frac{f(x+\delta) - f(x-\delta)}{2\delta}$$

and use $\hat{f}_\delta(x)$ as our estimate of $f'(x)$. The function `f.prime()` we ask you to write below implements $\hat{f}_\delta(x)$.

Write a function, `f.prime(x)`, to calculate the numerical derivative given the location x. Take the approximation window to be $\delta = 0.001$ (though it'd be nice to allow `f.prime()` to take this as an argument as well, with a default to 0.001). What is the output of `f.prime(-2)`?

```
# create a function for f'
f.prime <- function(x){

  # define delta
  delta = 0.001

  # cadd the delta values to the derivative function
  y.prime = (f(x + delta) - f(x - delta)) / (2*delta)

  # return the value
  return(y.prime)
}

# print the value for x = -2
f.prime(-2)
```

```
## [1] -0.1079819
```

The output of f.prime(-2) is *-0.1079819*

Question 3 [3 pts]

Write a function, `grad.des(x1)` to perform gradient descent from the starting point x1. Take the step sizes $\alpha_n = \frac{1}{n}$ and precision $\epsilon = 0.000001$, and let the maximum iterations be 10,000. That is compute:

$$x_{n+1} := x_n - \frac{1}{n}\hat{f}_\delta(x_n)$$

Your function should iterate through n until either the precision is satisfied or the max iteration is reached, whichever comes first. It should output a list, including the number of total iterations N, the minima $x^* = x_N$, the minimum value $f_{\min} = f(x^*)$ and the vectors of your search trajectory for both x and $f(x)$ values: $(x_1, x_2, \cdots, x_N)$ and $(f(x_1), f(x_2), \cdots, f(x_N))$.

```
set.seed(2022)

grad.des <- function(x1, precision = 0.000001, iterations = 10000){

  # set initial conditions of the function
  x_n = x1
  f_n = f(x_n)
  N = 1

  # set empty values for minimum of x and f(x)
  x_min = NA
  f_min = NA

  # create empty vectors for all x and f(x) values observed
  x_all = rep(NA,iterations)
  f_all = rep(NA,iterations)

  # use a while loop instead of a for loop because number of iterations is unknown
  # do the loop for as long as the absolute value of the slope is greater than the precision value and
  while ((abs(f.prime(x_n)) >= precision) & (N+1 <= iterations)) {

    # create vectors with all of the x and f(x) values observed in the iterations
    x_all[N] = x_min = x_n
    f_all[N] = f_min = f_n

    # calculate the x values and corresponding y values
    x_n = x_n - (1/N) * f.prime(x_n)
    f_n = f(x_n)

    # make n values increase
    N = N + 1

  }

  # merge all results together
  results = list(N = N,
                 x_min = x_min,
                 f_min = f_min,
```

```
                x_all = x_all,
                f_all = f_all)

  # return the results
  return(results)

}
```

Question 4 [2 pts]

Start from $x_1 = -2$, what is the minimum your function finds? Plot the points $(x_1, f(x_1)), (x_2, f(x_2)), \cdots, (x_N, f(x_N))$ in red with the curve of f from Question 1.

```
x <- grad.des(-2)


# print the x and f(x) minimums
x$x_min
```

```
## [1] -0.3324112
```

```
x$f_min
```

```
## [1] -0.3660469
```

The function finds a minimum at the point (`rx$x_min`,`rx$f_min`).

```
# load tidyverse
library(tidyverse)
```

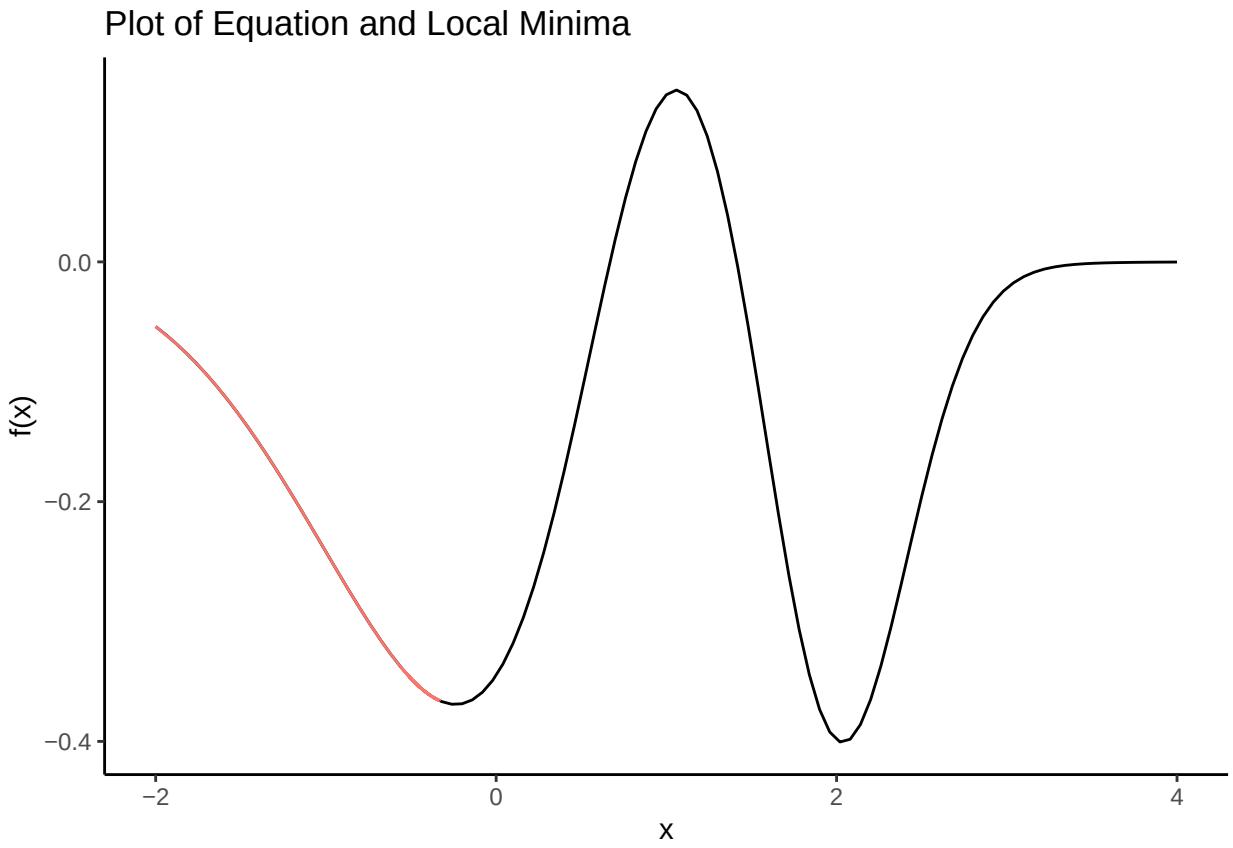
```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# plot the new values on the old curve - it was easier to use ggplot for this
ggplot() +
  geom_function(fun = f, xlim = c(-2,4)) +
  geom_line(aes(x$x_all,x$f_all, color = "red")) +
  theme_classic() +
  theme(legend.position = "none") +
  labs(x = "x", y = "f(x)", title = "Plot of Equation and Local Minima")
```

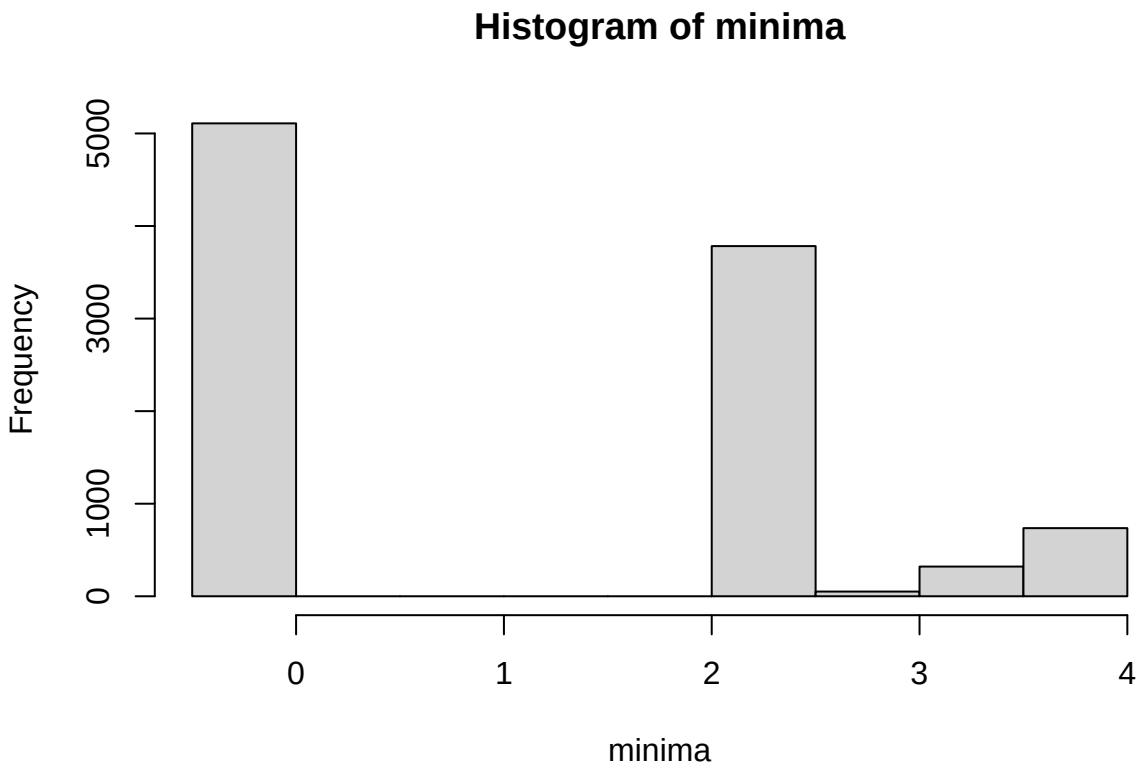
```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```



Question 5 [2 pts]

A gradient descent algorithm generally finds local minima, and these minima may not be global. One way to address this is to randomly start gradient descent at several different points and compare the function values at the resulting local minima. Generate 10,000 random starting points between $(-2, 4)$ (after setting seed with `set.seed(2022)`), and calculate the gradient descent result for each of them. Plot the histogram of the 10,000 minima you found. What is the global minimum? n[which(myfmin==min(myfmin))]

```
set.seed(2022)

start_val <- rep(NA,10000)
minima <- rep(NA,10000)

for (i in 1:10000) {

  start_val[i] <- runif(1, min=-2, max=4)

  minima[i] <- grad.des(start_val[i])$x_min

}


hist(minima)
```


Histogram of minima
Frequency
0
1000
3000
5000
0
1
2
3
4
minima