Gov 2018 Problem Set 1: Gradient Descent

Your name here:

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,

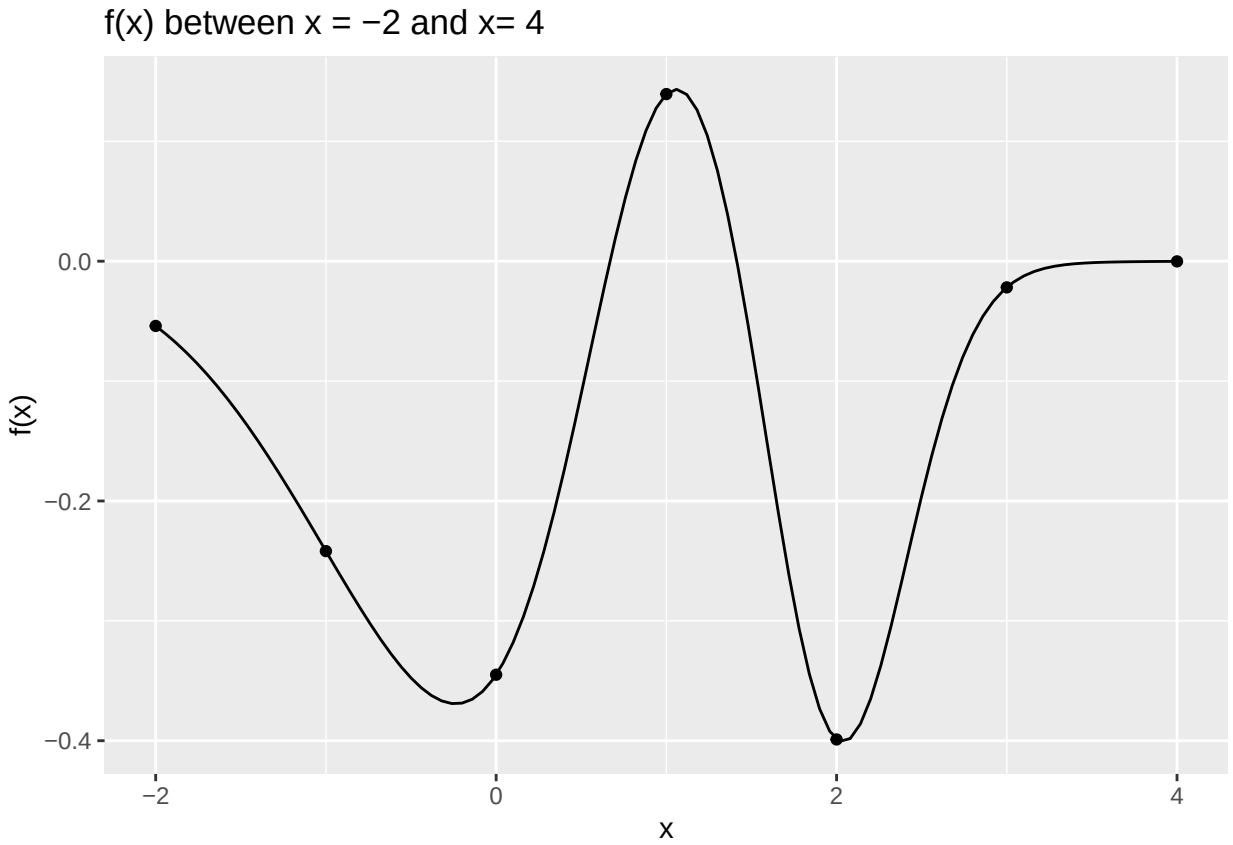
$$f(x) = -\frac{1}{\sqrt{2\pi}}\exp\big(-\frac{1}{2}x^2\big) + \frac{1}{\sqrt{2\pi}}\exp\big(-\frac{1}{2}(\frac{x-1}{0.5})^2\big) - \frac{1}{\sqrt{2\pi}}\exp\big(-\frac{1}{2}(\frac{x-2}{0.4})^2\big)$$

Question 1 [1 pt]

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

```
library(tidyverse)

# Coding f(x)
q1 <- function (x) {
  # Calculating the first term of the function
  first_term <- exp(-0.5 * (x^2))
  first_term <- (-1) * first_term * (1/(sqrt(2 * pi)))

  # Calculating the second term of the function
  second_term <- exp((-0.5) * (((x-1)/0.5)^2))
  second_term <- second_term * (1/(sqrt(2 * pi)))

  # Calculating the third term of the function
  third_term <- exp((-0.5)* (((x - 2)/0.4)^2))
  third_term <- (-1) * third_term * (1/(sqrt(2 * pi)))

  return(first_term + second_term + third_term)
}

# Getting all f(x) between x=-2 x=4
ys <- c()
for(i in -2:4) {
  x <- q1(i)
  ys <- c(ys, x)
}
```

```
fdata <- as_tibble(cbind(-2:4, ys))


# Plotting
fdata %>%
  ggplot(aes(x = V1, y = ys)) +
  geom_point() +
  geom_function(fun = q1) +
  labs(title = "f(x) between x = -2 and x= 4",
       x = "x",
       y = "f(x)")
```



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)`?

```
# Writing f'(x) function based on the derivative calculated above
f.prime <- function(x, delta) {
  return((q1((x + delta)) - q1((x - delta)))/(2 * delta))
}
```

```
f.prime(-2, 0.001)
```

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

f.prime(-2,0.001) outputs -0.108

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 $(f(x_1), f(x_2), \cdots, f(x_N))$.

```
# Gradient descent function
grad.des <- function(x1, max_iter) {

  # start n at 1
  n <- 1
  # Set current minimum at first x
  minima_x <- x1
  # Set current minimum at first y
  minima_y <- q1(x1)
  # To save path of x and f(x)
  x_trajec <- c(minima_x)
  fx_trajec <- c(minima_y)

  # Simulates max_iter times
  for(i in 0:max_iter) {
    # Calculate the step difference...
    step_difference <- (-1) * (1/n) * f.prime(x1, 0.001)
    # ... and apply it to x1 to get new x
    x1 <- x1 + step_difference
    # Plug in to get new y
    y1 <- q1(x1)
    # Save path of x and f(x)
    x_trajec <- c(x_trajec, x1)
    fx_trajec <- c(fx_trajec, y1)
    # Replace minimum if new minimum found
    if (y1 < minima_y) {
      minima_y <- y1
      minima_x <- x1
    }
    # Increase n
    n <- n + 1
    # Break loop if steps between x gets extremely small
    if (step_difference < 0.000001) {
      break
    }
  }
```

```
  return(list(n, minima_x, minima_y, x_trajec, fx_trajec))
}
```

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.

```
output <- grad.des(-2, 10000)
```

```
output[[2]]
```

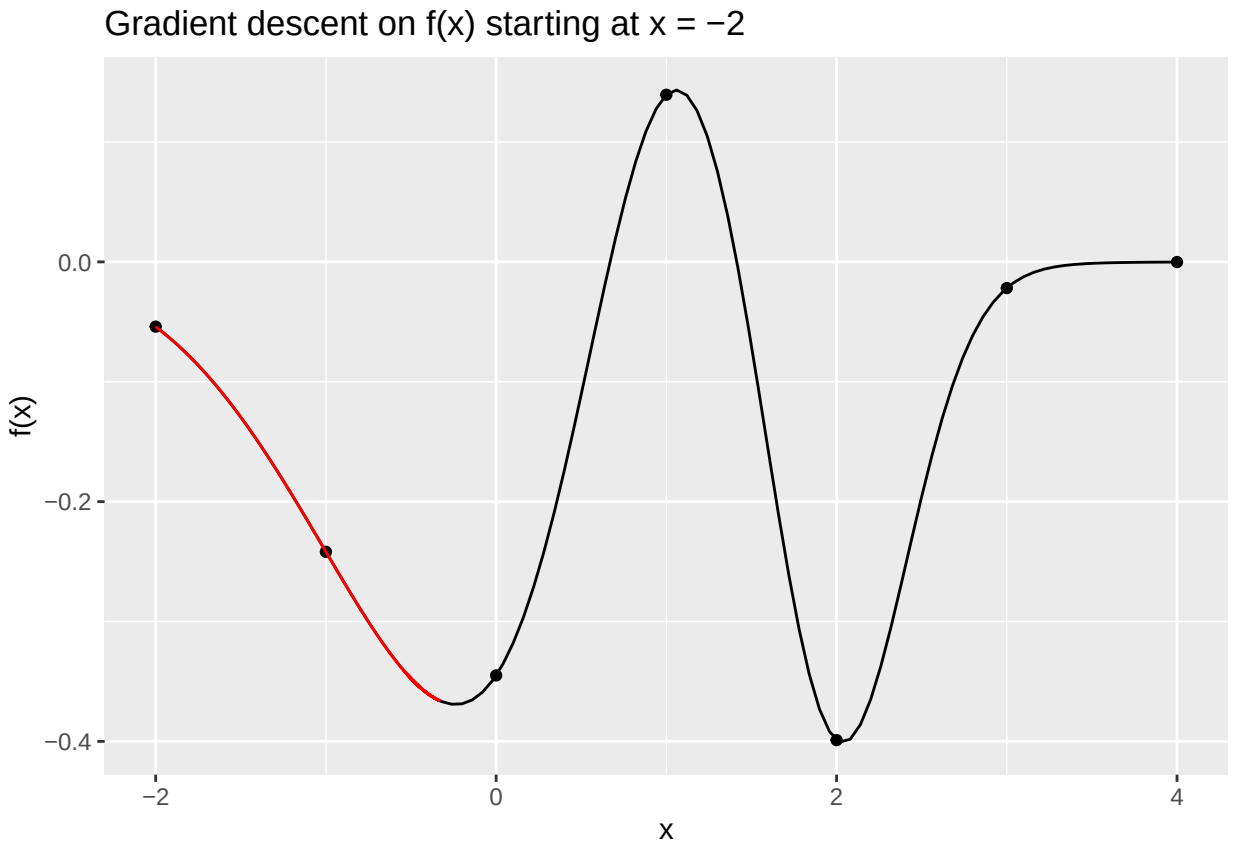
```
## [1] -0.3323919
```

```
output[[3]]
```

```
## [1] -0.3660482
```

Starting from -2, the minimum my function found was (-0.33, -0.367). Its path is plotted in red below.

```
# Plotting path of gradient descent function

dat <- tibble(x = output[[4]], y = output[[5]])

fdata %>%
  ggplot(aes(x = V1, y = ys)) +
  geom_point() +
  geom_function(fun = q1) +
  geom_line(data = dat, aes(x, y), color = "red") +
  labs(title = "Gradient descent on f(x) starting at x = -2",
       x = "x",
       y = "f(x)")
```

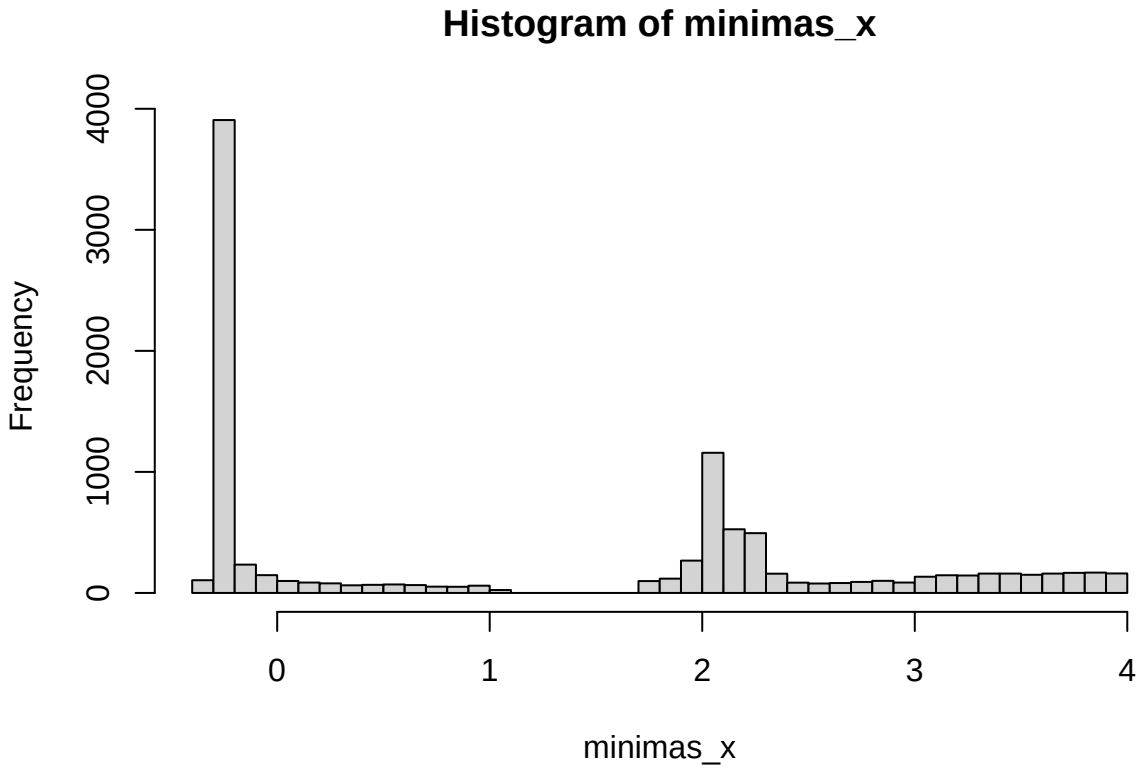


Question 5 [2 pts]

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)

global_desc <- function() {

  # Randomly finding 10k start points to search entire function
  start_points <- runif(10000, -2, 4)
  minima_x <- c()
  minima_y <- c()

  # Initiate gradient descents at each 10k start points
  for(i in start_points) {
    minima_x <- c(minima_x, grad.des(i, 10000)[[2]])
    minima_y <- c(minima_y, grad.des(i, 10000)[[3]])
  }

  return(list(minima_x, minima_y))
}

minimas <- global_desc()
```

```r
minimas_x <- minimas[[1]]
minimas_y <- minimas[[2]]

hist(minimas_x, breaks = 50)
```




```r
minimas <- cbind(minimas_x, minimas_y)

# Locating the true global minimum using the function given in the problem

global_min <- minimas[which(minimas[,2]==min(minimas[,2])), 1]
global_min
```

```
## minimas_x
##   2.03655
```

The global minimum is 2.03655. As you can see in the histogram, the gradient descent function came upon -0.33 and 2 most often, as they are the two most significant minima in the plot.