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,

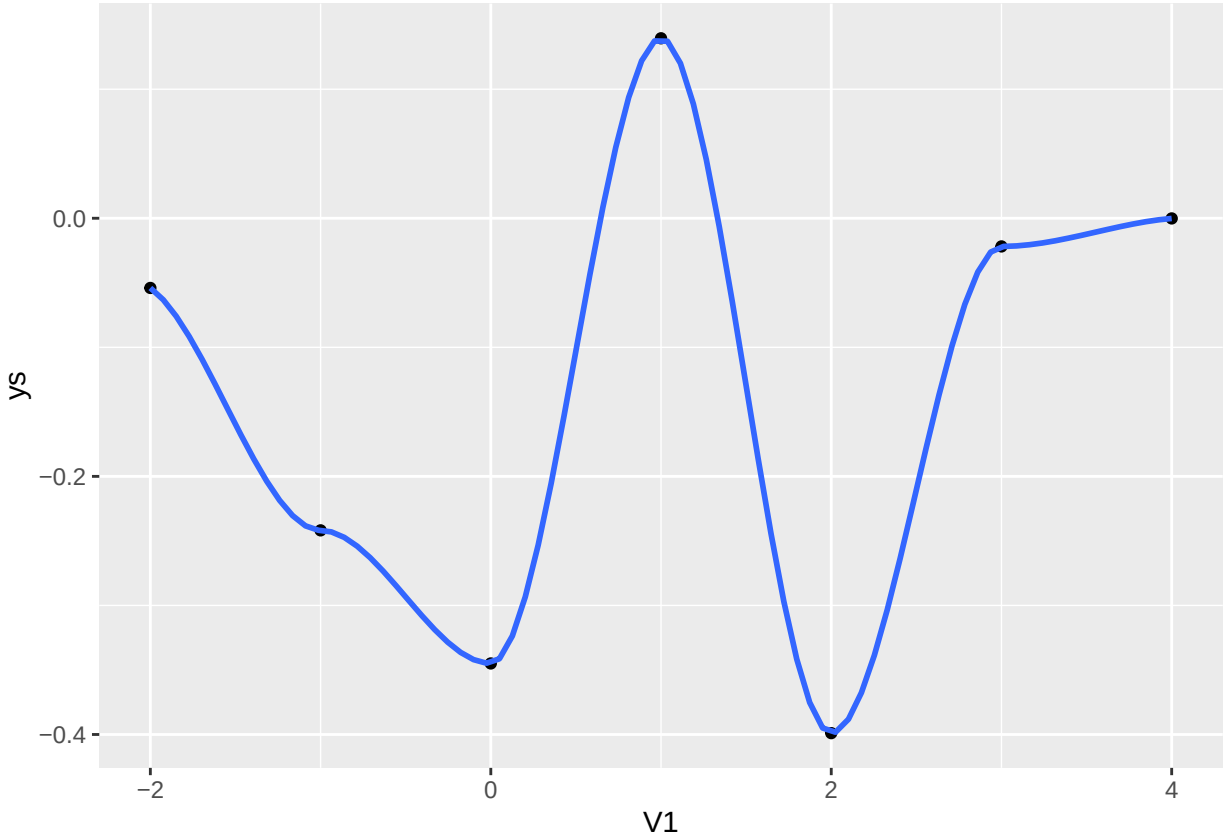
$$f(x) = -\frac{1}{\sqrt{2\pi}}\exp\big(-\frac{1}{2}x^2\big) + \frac{1}{\sqrt{2\pi}}\exp\big(-\frac{1}{2}(\frac{x-1}{0.5})^2\big) - \frac{1}{\sqrt{2\pi}}\exp\big(-\frac{1}{2}(\frac{x-2}{0.4})^2\big)$$

Question 1 [1 pt]

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

```
library(tidyverse)

q1 <- function (x) {
  first_term <- exp(-0.5 * (x^2))
  first_term <- (-1) * first_term * (1/(sqrt(2 * pi)))

  second_term <- exp((-0.5) * (((x-1)/0.5)^2))
  second_term <- second_term * (1/(sqrt(2 * pi)))

  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)
}

#for
ys <- c()
for(i in -2:4) {
  x <- q1(i)
  ys <- c(ys, x)
}

fdata <- as_tibble(cbind(-2:4, ys))
#fspline <- as.data.frame(spline(fdata$V1, fdata$y))

fdata %>%
```

```
  ggplot(aes(x = V1, y = ys)) +
  geom_point() +
  stat_smooth(method = 'loess', span = 0.5)
```



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)`?

```
f.prime <- function(x, delta) {
  return((q1((x + delta)) - q1((x - delta)))/(2 * delta))
}

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

```
## [1] -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 $(f(x_1), f(x_2), \cdots, f(x_N))$.

```
grad.des <- function(x1, max_iter) {
  n <- 1
  minima_x <- x1
  minima_y <- q1(x1)
  x_trajec <- c(minima_x)
  fx_trajec <- c(minima_y)

  for(i in 0:max_iter) {
    step_difference <- (-1) * (1/n) * f.prime(x1, 0.001)
    #print(step_difference)
    x1 <- x1 + step_difference
    y1 <- f.prime(x1, 0.001)
    x_trajec <- c(x_trajec, x1)
    fx_trajec <- c(fx_trajec, y1)
    if (y1 < minima_y) {
      minima_y <- y1
      minima_x <- x1
    }
    n <- n + 1
    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)
```

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

fdata %>%
  ggplot(aes(x = V1, y = ys)) +
  geom_point() +
  stat_smooth(method = 'loess', span = 0.5) +
  geom_line(data = dat, aes(x, y), color = "red")
```

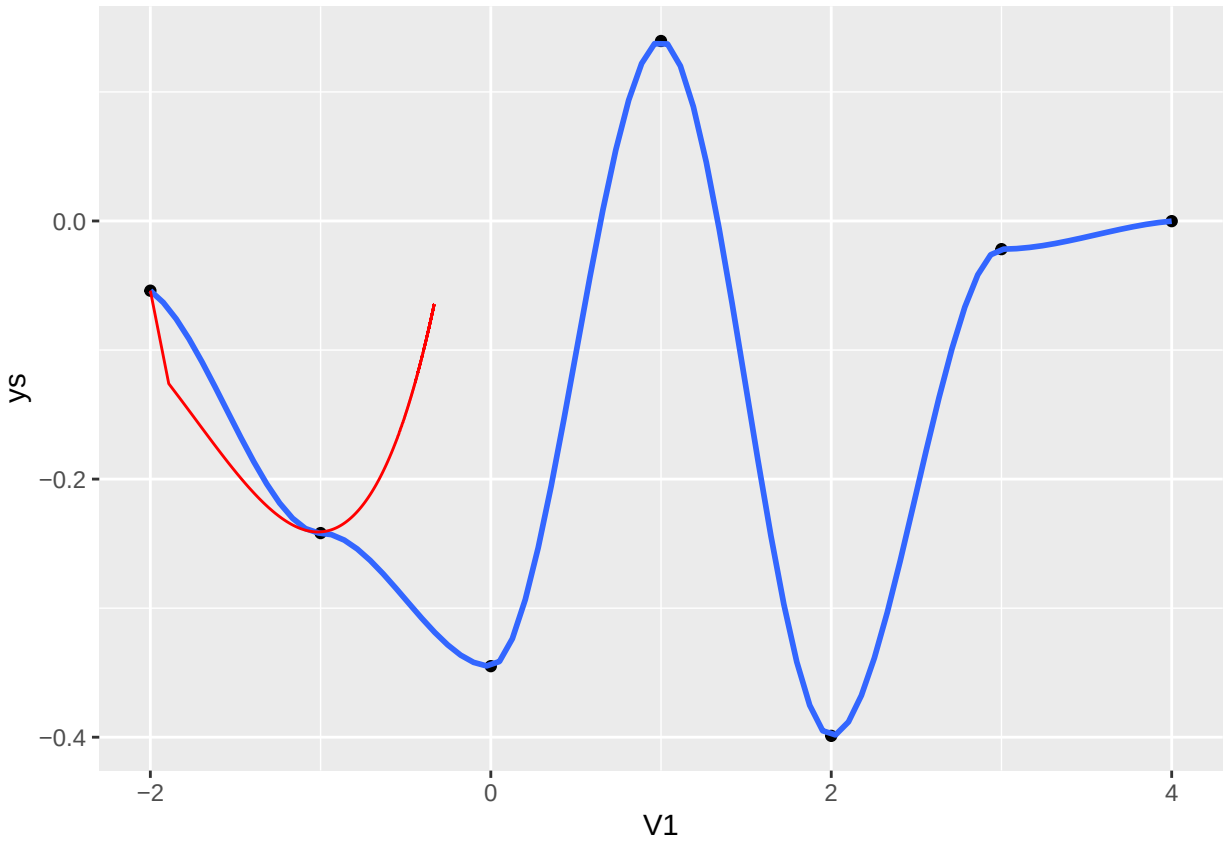
```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : span too small. fewer data values than degrees of freedom.
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -2.03
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 2.03
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 4.1209

## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : span too small. fewer
## data values than degrees of freedom.

## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at
## -2.03

## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius 2.03

## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition
## number 0

## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : There are other near
## singularities as well. 4.1209

## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



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() {

  start_points <- runif(10000, -2, 4)
  minima <- c()

  for(i in start_points) {

  }

}
```