HW 1

Bryan Mui - UID 506021334 - 14 April 2025

Problem 1

Part a

Find the theoretical min for the function:

$$f(x) = x^4 + 2x^2 + 1$$

Solution: find f'(x) and f' '(x), set f'(x) to 0 and solve, and f' '(x) needs to be > 0 to be a min

Step 1: find f'(x) and f' '(x)

$$\begin{align} f(x) &= x^4 + 2x^2 + 1 \\ f'(x) &= 4x^3 + 4x \\ f''(x) &= 12x^2 + 4 \\ & \end{align}$$

Step 2: set f'(x) to 0 and solve

$$\begin{align} f'(x) &= 4x^3 + 4x \\ 0 &= 4x^3 + 4x \\ 0 &= 4x(x^2 + 4) \end{align}$$

We get

$$x = 0$$

and

$$0 = x^2 + 4$$

which has no real solution

Step 3: check that f' '(x) needs to be > 0 to be a min

Our critical point is x = 0,

$$f''(0) = 12(0)^2 + 4 \tag{8}$$
$$= 4 \tag{9}$$

Since f'(x) = 0 at 0 and f' '(x) > 0 at that point, **we have a min at x = 0, and plugging into f(0) we get the minimum point**

(0, 1)

Part b

0)

Use the gradient descent algorithm with **constant step size** and with **back-tracking line search** to calculate x_min

Backtracking line search is implemented as follows:

1. Select a random starting point

2. While stopping criteria < tolerance, do:

- Select η_k(as a constant)

- Calculate $x_{(k+1)} = x_k - \eta_k * \nabla(f(x_k))$

- Calculate the value of stopping criterion

Stopping criteria: True if $\mid f(x_{k+1}) - f(x_k) \mid \leq \epsilon$

```
# Gradient descent algorithm that uses backtracking to minimize an objective
↪ function

gradient_descent_backtracking_constant_step <- function(tol = 1e-6, max_iter
↪ = 10000, step_size = 1, epsilon = 0.5, tau = 0.8) {
  # Initialize
  set.seed(777)
```

```
  iter <- 1
  step_size <- step_size
  max_iter <- max_iter
  tol <- tol
  epsilon <- epsilon
  tau <- tau
  obj_values <- numeric(max_iter)
  eta_values <- rep(step_size, max_iter)  # To store eta values used each
 ↪  iteration
  x0 <- runif(1, min=-10, max=10) # our first guess is somewhere between 0-1

  # Set the objective function to the function to be minimized
  # Objective function: f(x)
  obj_function <- function(x) {
    x^4 + 2*(x^2) + 1
  }

  # Gradient function
  gradient <- function(x) {
    4*x^3 + 4*x
  }

  for (iter in 1:max_iter) {

  }

  return(list(beta = beta, obj_values = obj_values, eta_values = eta_values))
   ↪  # in this case, beta(predictors) are the x values, obj_values are f(x),
   ↪  and eta is the step size
}
```

1) For the constant step size version of gradient descent, discuss how you selected the step size used in your code

Theoretical Analysis proves that for functions with a unique global minimum, the step size should be within 0 to 1/L to converge to the unique global minimum

2) For both versions of the gradient descent algorithm, plot the value of $f(x_k)$ as a function of k the number of iterations

3) or the the gradient descent method with backtracking line search, plot the step size η_k selected at step k as a function of k. Comment on the result