PHY1112: Assignment 12

> An extreme problem

Assigned: April 2th, 2024

Due: April 9th, 2024

Learning Objectives

1. Learn gradient descent for functions of a single variable.

Grade Breakdown

|  |  |  |
| --- | --- | --- |
| Part | 1 | Total |
| Points | 24 | 24 |
| Score |  |  |

**Question 1: Gradually to the Extreme.**

In this question, we will apply gradient descent for functions of a single variable to find the minimum of a quartic function given by:

1. Write a Python function that returns the value of of the quartic function given above for an input value of .   
   **(2 marks)**
2. Write a Python function that performs gradient descent to find the minimum of a function of one variable

The function should take as inputs:

* Function handle for
* Starting point for the gradient descent,
* Step size for numerical derivative,
* The descent rate
* The convergence condition
* The maximum number of iterations,

The output should be:

* The found minimum

The gradient descent algorithm should implement the numerical derivative as a central difference.

The code should test the following convergence condition: |

The function should print a warning message if the maximum number of iterations is reached without satisfying the convergence condition. In this case it should return nan or none.  
**(10 marks)**

1. Using your gradient descent function from part b), find the minimum of the quartic function defined above, for initial positions , ,   
   **(3 marks)**

**A black screen with white text

Description automatically generated**

1. Some of your answers in part ‘c’ give different results. Why? Which answer is the global minimum? How do you know? Plot the function. Use this plot to assist with your explanation.  
   **(5 marks)**

Different results are achieved at x = 1 as opposed to x = 0 and x = -1 because there are two local minima in the graph, one being closer to x = 1, the other being closer to x = 0 and x = -1. As observed from the graph, the minima at x ≈ -1.3 is the global minima.

A graph of a function

Description automatically generated

**Figure 1.** A graph of the function in the domain of [-2, 2].

1. What single input can you alter in order to make your gradient *descent* function find the maxima instead (that is, it performs gradient *ascent*)? Why does this happen?  
     
   Change this input, and find the local maximum between your two minima from part ‘c’  
   **(3 marks)**

By changing whether the gradient product is added or subtracted from the initial x, we can change whether the function is a gradient ascent or descent. This happens because that’s how math works. . . (am I missing something?).

A black screen with white text

Description automatically generated

**(24 marks total, 1 for docstrings/file header/variable naming/comments)**

**CODE:**

'''

Filename:       assignment\_12.py

Author:         Patrick Geraghty

Date Created:   2024-04-08

Date Modified:  2024-04-08

Description:    Contains the functions quartic\_function, gradient\_descent, plot\_quartic\_function, gradient\_ascent, and main. Upon execution, main tests the gradient\_descent function and the gradient\_ascent function using provided initial x positions and a graph for comparison.

'''

import numpy as np

import matplotlib.pyplot as plt

*def* quartic\_function(*x*):

    '''

    (float) -> float

    This function returns the value of the quartic function x^4 - 3x^2 + x + 1 at x.

    Preconditions: x is a number

    '''

    return x\*\*4 - 3\*x\*\*2 + x + 1

*def* gradient\_descent(*f*, *x*, *h*, *gamma*, *tol*, *max\_iter*):

    '''

    (function, float, float, float, float, int) -> float

    This function finds the minimum of a function using the gradient descent method for a single variable function.

    Preconditions: f is a single variable function, tol > 0, max\_iter > 0

    '''

    for \_ in range(max\_iter):                                       # iterate through the maximum number of iterations

        grad = (f(x + h / 2) - f(x - h / 2)) / h                                # calculate the gradient for f(x) at x

        x -= gamma \* grad                                           # update x

        # if np.linalg.norm(grad) < tol:                            # if the norm of the gradient is less than the tolerance, break (method 1)

        if np.linalg.norm(f(x) - f(x + gamma \* grad)) < tol:        # if the difference between f(x\_new) and f(x\_current) is less than the tolerance, break (method 2)

            break

        if \_ == max\_iter - 1:                                       # if the maximum number of iterations is reached, print a message and return None

            print("Max iterations reached")

            return None

    return x                                                        # return the minimum of the function

*def* plot\_quartic\_function():

    '''

    () -> None

    This function plots the quartic function x^4 - 3x^2 + x + 1.

    Preconditions: None

    '''

    x = np.linspace(-2, 2, 1000)                                    # create an array of x values within the necessary bounds

    y = quartic\_function(x)                                         # create an array of y values for the quartic function

    plt.figure()                                                    # create a new figure

    plt.plot(x, y, *label*='f(x)')                                    # plot the quartic function

    plt.xlabel('x')                                                 # label the x-axis

    plt.ylabel('y')                                                 # label the y-axis

    plt.title('Quartic Function')                                   # title the plot

    plt.grid()                                                      # add a grid to the plot

    plt.legend()                                                    # add a legend to the plot

    plt.show()                                                      # show the plot

*def* gradient\_ascent(*f*, *x*, *h*, *gamma*, *tol*, *max\_iter*):

    '''

    (function, float, float, float, float, int) -> float

    This function finds the maximum of a function using the gradient ascent method for a single variable function.

    Preconditions: f is a single variable function, tol > 0, max\_iter > 0

    '''

    for \_ in range(max\_iter):                                       # iterate through the maximum number of iterations

        grad = (f(x + h / 2) - f(x - h / 2)) / h                                # calculate the gradient for f(x) at x

        x += gamma \* grad                                           # update x

        # if np.linalg.norm(grad) < tol:                            # if the norm of the gradient is less than the tolerance, break (method 1)

        if np.linalg.norm(f(x) - f(x + gamma \* grad)) < tol:        # if the difference between f(x\_new) and f(x\_current) is less than the tolerance, break (method 2)

            break

        if \_ == max\_iter - 1:                                       # if the maximum number of iterations is reached, print a message and return None

            print("Max iterations reached")

            return None

    return x                                                        # return the maximum of the function

*def* main():

    '''

    () -> None

    This function tests the gradient\_descent function and the gradient\_ascent function using provided initial x positions and a graph for comparison.

    Preconditions: None

    '''

    print(*f*'The minimum of f(x) from initial position x = 1 is: x = {gradient\_descent(quartic\_function, 1, 0.01, 0.1, 1e-6, 1000)}')

    print()

    print(*f*'The minimum of f(x) from initial position x = 0 is: x = {gradient\_descent(quartic\_function, 0, 0.01, 0.1, 1e-6, 1000)}')

    print()

    print(*f*'The minimum of f(x) from initial position x = -1 is: x = {gradient\_descent(quartic\_function, -1, 0.01, 0.1, 1e-6, 1000)}')

    print()

    plot\_quartic\_function()

    print()

    print(*f*'The maximum of f(x) from initial position x = 1 is: x = {gradient\_ascent(quartic\_function, 1, 0.01, 0.1, 1e-6, 1000)}')

    print()

    print(*f*'The maximum of f(x) from initial position x = 0 is: x = {gradient\_ascent(quartic\_function, 0, 0.01, 0.1, 1e-6, 1000)}')

    print()

    print(*f*'The maximum of f(x) from initial position x = -1 is: x = {gradient\_ascent(quartic\_function, -1, 0.01, 0.1, 1e-6, 1000)}')

main()