


Practical No. 4: Manual Content

	Guru Gobind Singh Foundation's Guru Gobind Singh College of Engineering and Research Center, Nashik				
Experiment No: 04					
Title of Experiment: Implement Gradient Descent Algorithm to find the local minima of a function. For example, find the local minima of the function $y=(x+3)^2$ starting from the point $x=2$.					
Student Name:					
Class:	BE (Computer)				
Div:	-	Batch:	CO		
Roll No.:					
Date of Attendance (Performance):					
Date of Evaluation:					
Marks (Grade) Attainment of CO Marks out of 10	A	P	W	T	Total
CO Mapped	CO3: Select and apply appropriately supervised machine learning algorithms for real time applications.				
Signature of Subject Teacher					

TITLE: Implement Gradient Descent Algorithm to find the local minima of a function. For example, find the local minima of the function $y=(x+3)^2$ starting from the point $x=2$.

AIM: The aim of this practical is to illustrate the implementation of the Gradient Descent algorithm for locating local minima of a given function. This involves understanding the gradient descent method, performing iterations to converge towards a local minimum, and evaluating the effectiveness of the algorithm in the context of the function $y = (x + 3)^2$ starting from the initial point ($x = 2$).

OBJECTIVES: Based on above main aim following are the objectives

1. **Understanding Optimization Techniques:** Develop a solid understanding of optimization methods, with a focus on the Gradient Descent algorithm. Grasp the concept of finding local minima in mathematical functions through iterative updates.
2. **Algorithm Implementation:** Implement the Gradient Descent algorithm practically. Apply it to the function $y = (x + 3)^2$, beginning with an initial value of ($x = 2$), and observe how the algorithm converges towards a local minimum.
3. **Analyzing Convergence:** Examine the iterative behavior of the Gradient Descent algorithm. Gain insights into how the algorithm iteratively refines the value of (x) by utilizing the gradient of the function.
4. **Algorithm Evaluation:** Evaluate the effectiveness of the Gradient Descent algorithm in successfully identifying the local minimum of the given function. Compare the obtained result with the anticipated theoretical solution.

Prerequisite:

Basic of Python, Concept of Gradient Descent Algorithm

Software Requirements:

Anaconda with Python 3.7

Hardware Requirement:

PIV, 2GB RAM, 500 GB HDD

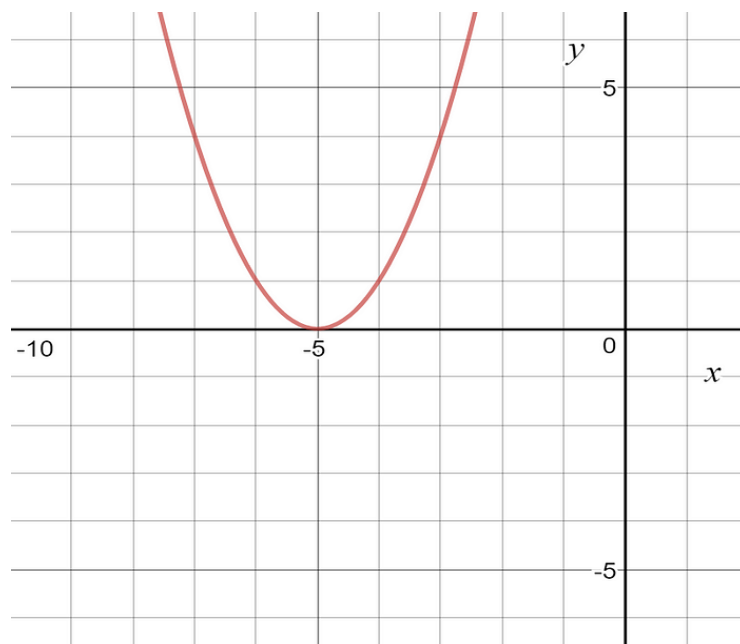
Theory Concepts:

What is gradient descent ?

It is an optimization algorithm to find the minimum of a function. We start with a random point on the function and move in the **negative direction** of the **gradient of the function** to reach the **local/global minima**.

Example:

Question : Find the local minima of the function $y=(x+5)^2$ starting from the point $x=3$



Solution : We know the answer just by looking at the graph. $y = (x+5)^2$ reaches its minimum value when $x = -5$ (i.e when $x=-5$, $y=0$). Hence $x=-5$ is the local and global minima of the function.

Now, let's see how to obtain the same numerically using gradient descent.

Step 1 : Initialize $x = 3$. Then, find the gradient of the function, $dy/dx = 2*(x+5)$.

Step 2 : Move in the direction of the negative of the gradient ([Why?](#)). But wait, how much to move?

For that, we require a learning rate. Let us assume the **learning rate** $\rightarrow 0.01$

Step 3 : Let's perform 2 iterations of gradient descent

Initialize Parameters :

$$X_0 = 3$$

$$\text{Learning rate} = 0.01$$

$$\frac{dy}{dx} = \frac{d}{dx} (x + 5)^2 = 2 * (x + 5)$$

Iteration 1 :

$$X_1 = X_0 - (\text{learning rate}) * \left(\frac{dy}{dx}\right)$$

$$X_1 = 3 - (0.01) * (2 * (3 + 5)) = 2.84$$

Iteration 2 :

$$X_2 = X_1 - (\text{learning rate}) * \left(\frac{dy}{dx}\right)$$

$$X_2 = 2.84 - (0.01) * (2 * (2.84 + 5)) = 2.6832$$

Step 4 : We can observe that the X value is slowly decreasing and should converge to -5 (the local minima).

Implementing Gradient Descent in Python

Importing libraries

```
import numpy as np
import matplotlib.pyplot as plt
```

We then define the function f(x) and its derivative f'(x) –

```
def f(x):
    return (x+3)**2

def df(x):
    return 2*x + 6
```

$F(x)$ is the function that has to be decreased, and df is its derivative (x). The gradient descent approach makes use of the derivative to guide itself toward the minimum by revealing the function's slope along the way.

The gradient descent function is then defined.

```
def gradient_descent(initial_x, learning_rate, num_iterations):
    x = initial_x
    x_history = [x]

    for i in range(num_iterations):
        gradient = df(x)
        x = x - learning_rate * gradient
        x_history.append(x)

    return x, x_history
```

The starting value of x , the learning rate, and the desired number of iterations are sent to the gradient descent function. In order to save the values of x after each iteration, it initializes x to its original value and generates an empty list. The method then executes the gradient descent for the provided number of iterations, changing x every iteration in accordance with the equation $x = x - \text{learning_rate} * \text{gradient}$. The function produces a list of every iteration's x values together with the final value of x .

The gradient descent function may now be used to locate the local minimum of $f(x)$ –

```
initial_x = 2
learning_rate = 0.1
num_iterations = 50

x, x_history = gradient_descent(initial_x, learning_rate, num_iterations)

print("Local minimum: {:.2f}".format(x))
```

Output

Local minimum: -3.00

In this illustration, x is set to 2 at the beginning, with a learning rate of 0.1, and 50 iterations are run. Finally, we publish the value of x , which ought to be close to the local minimum at $x=-3$.

Plotting the function $f(x)$ and the values of x for each iteration allows us to see the gradient descent process in action –

```

#Create a range of x values to plot
x_vals = np.linspace(-1, 5, 100)

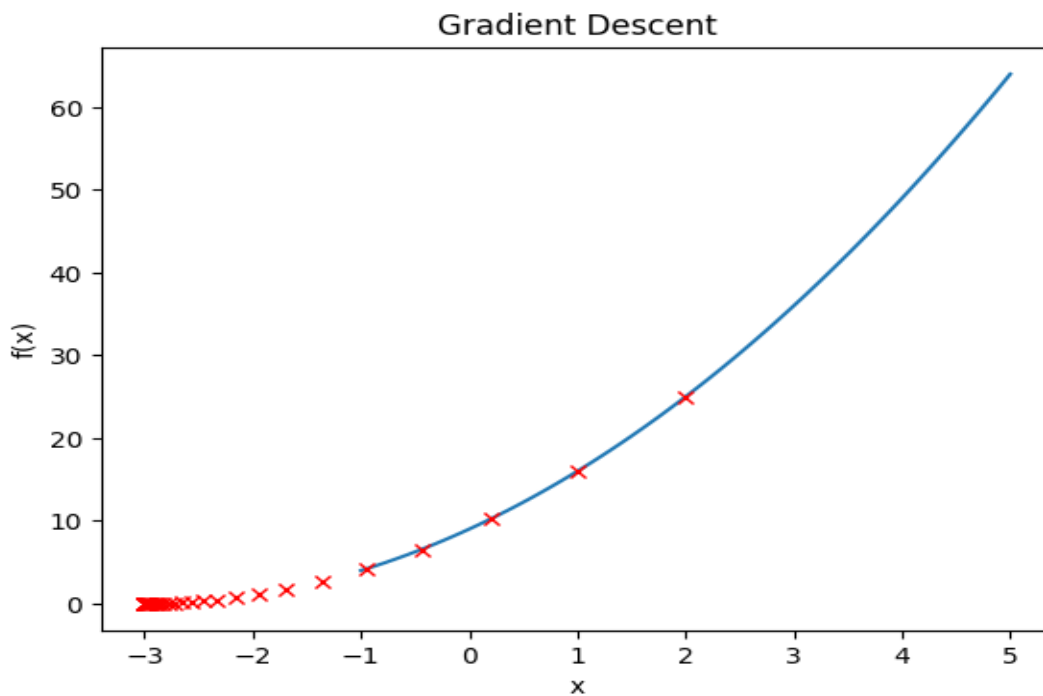
#Plot the function f(x)
plt.plot(x_vals, f(x_vals))

# Plot the values of x at each iteration
plt.plot(x_history, f(np.array(x_history)), 'rx')

#Label the axes and add a title
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Gradient Descent')

#Show the plot
plt.show()

```



Conclusion: Thus we have implemented Gradient Descent Algorithm to find the local minima of a function $y=(x+3)^2$ starting from the point $x=2$ that is -3 .