Labwork 1: Gradient Descent

April 14, 2024

1 Introduction

Gradient descent is an optimization technique frequently employed to train machine learning models and neural networks. It does this by minimizing the discrepancies between the model's predictions and the actual observed results. This labwork will implement the gradient descent from scratch in Python and evaluate it.

2 Implementation

- First, a f(x) function is implemented to calculate the value of $y = x^2$: the input parameter is x and the output value is x^2.
- Then, a derivative(x) function is implemented to calculate the value of f'(x), which is equal to $2 * x$: the input parameter is x and the output value is 2*x.
- Finally, a gradient_descent function is implemented to find the minimum of a function by iteratively updating the value of the variable x in the direction opposite to the gradient of the function.

- It takes four parameters: derivative_func - the derivative(x), initial_x - the initial value of the variable x to start the optimization process, learning_rate - the step size or the rate at which the algorithm updates the value of x in each iteration, num_iterations - the number of iterations the algorithm will perform.
- For the initialization, the initial value of x is assigned to the initial_x parameter.
- Then, the loop is run in num_iterations times. In each iteration, the derivative of the function is calculated by calling derivative_func(x), and the result is stored in the grad variable. The new value of x is calculated as x_new = x - learning_rate * grad. This is the core of the gradient descent algorithm, where the value of x is updated in the direction opposite to the gradient, scaled by the learning_rate. After that, the value of x is updated to x_new for the next iteration.

3 Evaluation

Test with $y = x^2$
With the number of iterations = 50, the initial x = 10, learning rate = 0.1, there are some findings:

- The value of f(x) decreases with each iteration, which is a good sign that the gradient descent is working correctly. After 50 iterations, the new f(x) is 2.0370359763344873e-08.
- The rate at which (f(x)) and x decrease is stable over iterations. This example shows a stable rate of eighty per cent.

4 Conclusion

In this labwork, I have completed the implementation and evaluation of gradient descent in Python. An interesting finding is that the decreasing rates of f(x) and x are stable after iterations.