Labwork 1: Gradient Descent

Le Chi Thanh - M23.ICT.011

04/13/24

1 Objective

The primary objective of this labwork is to implement a gradient descent algorithm from scratch to find the minimum value of a given function.
The gradient descent algorithm is an optimization technique used in machine learning for finding local minima of functions. In the scope of labwork, we will apply the algorithm to the function f(x) = x^2 and its first-order derivative f2(x) = 2.x to understand the implementation and observe the convergence process.
The process begins with initializing a starting point and iteratively adjusting it using the gradient (derivative) of the function at each step. The learning rate, a key parameter, controls the value of each update and thus influences the speed and success of convergence. During the process, we will print the intermediate iterative steps, including the iteration count, execution time, current value of x, and the corresponding function value f(x).
Finally, we discuss the working of gradient descent, examine how different learning rates affect the optimization process, and observe how the algorithm converges to the minimum value of the function.

2 Implementation of the Algorithm

1. Function
For this lab, the function f(x) = x^2 is chosen, and its derivative is f2(x) = 2.x . This function has a global minimum at x=0 and f(x)=0.
2. Inintialization

The algorithm starts with an initial point `starting_point`, which is set to 10.0 in this case. A learning rate (`learning_rate`) of 0.1 is chosen, which dictates the step size in each iteration. A threshold (`threshold`) of 10^{-6} is set to determine when to stop the algorithm when the change in function value becomes very small. A maximum step limit of 100 iterations (`max_step`) is chosen to prevent excessive computation.

3. Iterative Process

At each iteration, the current value of x, the function value f(x), and the derivative f2(x) are calculated. We also record the execution time of each iteration.

The new value of x is calculated using the gradient descent formula: x = x - `learning_rate` * f'(x).

The process stops if either the maximum step is reached or the absolute value of the derivative falls below the threshold, indicating convergence.

4. Result

As we observe, the algorithm covers rapidly, from the starting point of x=10.0, we quickly approach the minimum value at x=0 and f(x)=0. The printed result shows how the function value decreases and x approaches zero.

5. Analysis the Learning Rate

We examine the impact of Learning Rate by observing the process with different Learning Rate values. In the scope of this report, we examine with value ranging between 0.1,0.2,0.3,0.4,0.5,0.6,1.

Table 1: Table 5.1. Learning Rate Examination f(x) = x^2

Learning Rate	Total Time	Iterations
0.1	0.0003015995	76
0.2	0.0001270771	33
0.3	0.0001270771	19
0.4	0.0000948906 2	11
0.5	0.0000510216	1
0.6	N/A	Divergence

In all examinations, x and f(x) are both approximately equal to 0. We observe that a higher learning rate results in a shorter total execution time and fewer iterations required. However, to high learning rate would lead to the divergence of the result.

3 Evaluation

We experiment with the Gradient Descent algorithm with a new function and observe the result: f(x) = x^4. In this experiment the threshold is 10^{-2}

Table 2: Table 5.2. Learning Rate Examination f(x) = x^4

Learning Rate	Total Time	x	f(x)	Iterations
0.01	N/A	N/A	N/A	Divergence
0.005	N/A	N/A	N/A	Divergence
0.002	0.0138	0.1357	0.0003	3775
0.001	0.0247	0.1357	0.0003	6778
0.0001	∞	0.3533	0.0156	∞

We observe that as the learning rate becomes too high (from 0.005), the algorithm leads to the divergence of the result. 0.001 and 0.002 are good learning rates for the function to converge to optimal minimal results. We also acknowledge the saturated result at a learning rate equal to 0.0001, as the learning is low, the algorithms were unable to reach the real minimal values of the function.

4 Conclusion

In conclusion, the gradient descent algorithm is a useful optimization method that can be used to find the minimal value of functions. The algorithm's success depends significantly on the choice of the learning rate, which controls the iteration times and the updated values in each iteration. The choice of learning rate can have the following impact on the performance of the algorithm:

Learning Rate Too High:

If the learning rate is set too high, the algorithm may make unnecessary large changes for function value in each iteration. This can lead to divergence, where x continues to increase without bounds and the function value escalates. As a result, the updates may be too aggressive, and the algorithm may fail to converge.

Learning Rate Too Low:

If the learning rate is set too low, the algorithm may take a very small change for function value in each iteration.

This can result in a slow convergence, taking many iterations to reach the minimum. Although the algorithm is likely to eventually converge, it will do so at a slower pace, making the optimization process less efficient.
Finding the right learning rate is essential for achieving efficient convergence. Often, a moderate learning rate that balances the trade-off between convergence speed and stability is desirable. Too high or too low a learning rate could lead to a divergence or saturation state (in which algorithms are unable to reach to true minimal value of the function).