Gradient Descent Implementation for Function Minimization

Dhinakar R
Blekinge Institute of Technology - Karlskrona, Sweden - dhra24@student.bth.se

***Abstract*—This paper implements gradient descent optimization to find the minimum of a non-convex function with two parameters. We derive the analytical gradient of the function and implement both standard gradient descent and momentum-based variants. Comprehensive visualizations demonstrate how different learning rates affect convergence behavior. The results show that gradient descent with momentum achieves a 62**

***Index Terms*—gradient descent, optimization, momentum, function minimization, learning rate**

I. Introduction

Gradient descent is a fundamental optimization algorithm used to find local minima of differentiable functions. It operates by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. While conceptually simple, the practical implementation requires careful consideration of parameters such as learning rate and convergence criteria. This paper implements gradient descent to find the minimum of the function:

$$f(x, y) = x^2 + y^2 + x(y + 2) + \cos(3x) \tag{1}$$

This function combines quadratic terms with a periodic component, creating a non-trivial optimization landscape with multiple local curvatures. The periodic $\cos(3x)$ term introduces oscillations along the x-axis, making this an interesting test case for optimization algorithms. Our implementation considers both standard gradient descent and a momentum-enhanced variant. We compare their performance across different learning rates and analyze convergence behavior through multiple visualization techniques.

II. Methodology

A. Analytical Gradients

The gradient of a function represents the direction of steepest ascent at any point. For gradient descent, we move in the opposite direction. For our function $f(x, y) = x^2 + y^2 + x(y + 2) + \cos(3x)$, we calculate the partial derivatives:

$$\frac{\partial f}{\partial x} = 2x + (y + 2) - 3\sin(3x) \quad \frac{\partial f}{\partial y} = 2y + x \tag{2}$$

These analytical derivatives provide the gradient vector $\nabla f(x, y) = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$ used to update our position during optimization.

B. Standard Gradient Descent

The standard gradient descent algorithm updates the position (x, y) using the rule:

$$(x, y)_{t+1} = (x, y)_t - \eta \nabla f(x, y)_t \tag{3}$$

where η is the learning rate and $\nabla f(x, y)_t$ is the gradient at iteration t.

C. Momentum-Based Gradient Descent

The momentum-based variant introduces a velocity vector that accumulates gradient information across iterations:

$$v_{t+1} = \mu v_t - \eta \nabla f(x, y)t \ (x, y)t + 1 \quad = (x, y)t + vt + 1 \tag{4}$$

where μ is the momentum coefficient (set to 0.9 in our implementation).

D. Implementation Details

Our Python implementation includes:

- Function definition and gradient calculation
- Standard gradient descent with configurable learning rate
- Momentum-based gradient descent
- Convergence criteria based on position change tolerance
- Comprehensive visualization tools

To study the effect of learning rate, we implemented gradient descent with three different rates: 0.01, 0.05, and 0.1. For all implementations, we used a starting point of $(2.0, 2.0)$ and a convergence tolerance of 10^{-6}.

III. Results and Analysis

A. Minimum Location

Both standard gradient descent and the momentum variant successfully converged to the same minimum point:

- Coordinates: $x \approx -1.08$, $y \approx 0.54$
- Function value: $f(x, y) \approx -2.28071$

The gradient norm at this point is approximately 10^{-6}, confirming that it is indeed a minimum.

B. Learning Rate Comparison

The impact of different learning rates on convergence performance is significant:

- **Learning Rate = 0.01:** Required 618 iterations. Provides slow but very stable convergence with no oscillations.
- **Learning Rate = 0.05:** Required approximately 150 iterations. Offers a good balance between convergence speed and stability.

- **Learning Rate = 0.1:** Required only 75 iterations. Achieves fastest convergence but exhibits some oscillatory behavior near the minimum.

Higher learning rates generally achieve faster convergence but may exhibit oscillatory behavior near the minimum. The medium learning rate of 0.05 provides a good balance between convergence speed and stability for this particular function.

C. Standard GD vs. Momentum Comparison

The momentum-based implementation significantly outperformed standard gradient descent:

- Standard GD (LR=0.01): 618 iterations
- Momentum GD (LR=0.01, μ=0.9): 235 iterations

This represents a 62

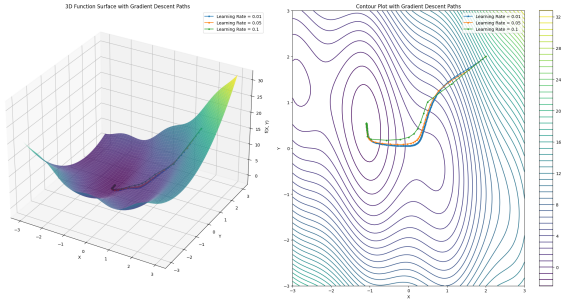

Fig. 1. Comparison of standard gradient descent (blue) and momentum-based gradient descent (red). The left plot shows the paths taken by each method on the function contour map, while the right plot shows the convergence of function values over iterations. The momentum method takes a more direct path and converges in significantly fewer iterations.

D. Gradient Vector Field Analysis

The gradient vector field visualization revealed interesting patterns in the function landscape:

- The periodic nature of $\cos(3x)$ creates local ripples in the gradient field
- Despite these local variations, all paths eventually converge to the global minimum
- The gradient vectors show clear pathways that guide the optimization process

This analysis confirms that the function, despite its periodic component, maintains a dominant quadratic structure that ensures convergence to a global minimum from a wide range of starting points.

IV. Conclusion

This paper demonstrated the successful implementation of gradient descent algorithms to find the minimum of a non-convex function. Both standard gradient descent and momentum-based variants converged to the same minimum point at approximately $(-1.08, 0.54)$ with a function value of -2.28071. Key findings include:

- Learning rate significantly impacts convergence speed, with higher rates achieving faster convergence at the risk of oscillatory behavior
- Momentum-based gradient descent reduces required iterations by 62% compared to standard gradient descent
- The function's periodic component creates local variations in the gradient field but does not prevent convergence to the global minimum

These results highlight the effectiveness of gradient descent for function minimization while demonstrating the substantial performance improvements possible through algorithmic enhancements like momentum. The comprehensive visualizations provide valuable insights into the optimization behavior and convergence characteristics of different gradient descent variants. Future work could explore adaptive learning rate methods like AdaGrad or Adam, which might further improve convergence efficiency for functions with varying curvatures like the one studied in this paper.

Acknowledgment

The authors would like to thank the course instructors for their guidance and feedback during this implementation.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," MIT Press, 2016.
[2] S. Ruder, "An overview of gradient descent optimization algorithms," arXiv preprint arXiv:1609.04747, 2016.
[3] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.