Gradient Descent Optimisation of a Bi-Variate Function

Sohan Arun
dept. Computer Science
Blekinge Institute of Technology
Karlskrona, Sweden
soar24@student.bth.se

***Abstract*—This study examines how gradient descent navigates the surface of a two-variable function to reach its local minimum. Through the derivation of exact closed-form gradients, a Python-based gradient-descent algorithm is implemented, starting at random point $(1.0, 1.0)$ with a learning rate of 0.1. The optimizer converges in only 71 iterations to the point $(x^\star, y^\star) = (-1.088162, 0.544077)$, yielding $f(x^\star, y^\star) = -2.280710$. Convergence behaviour is analysed, the impact of learning-rate and tolerance selection is examined, and the optimization trajectory is showcased via a three-dimensional surface plot.**

***Index Terms*—Gradient descent, non-linear optimisation, Python**

I. Problem Formulation

The study addresses the optimisation of the two-variable function

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

with the following four objectives:

1) Derive analytic expressions for the partial derivatives $\dfrac{\partial f}{\partial x}$ and $\dfrac{\partial f}{\partial y}$.
2) Implement a straightforward gradient-descent algorithm that uses these derivatives to iteratively update the pair (x, y) toward a stationary point.
3) Demonstrate convergence to the global minimum, reporting both its coordinates $\left(x^\star, y^\star\right)$ and the corresponding function value $f(x^\star, y^\star)$.
4) Visualise the optimisation trajectory on the surface of $f(x, y)$.

II. Analytic Derivation of the Gradient

The first-order partial derivatives of the objective function in (1) are

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

$$\frac{\partial f}{\partial y} = x + 2y. \tag{3}$$

Each expression was confirmed symbolically using `SymPy`.

III. Gradient Descent Implementation

The analytic gradients from (2)–(3) drive the gradient-descent routine. The algorithm appears in Algorithm III-A, followed by hyper-parameters and stopping criterion.

A. Pseudocode

Require: initial point (x_0, y_0), step size η, tolerance ε, maximum iterations $K_{\max}$
Ensure: final iterate (x_K, y_K) and trajectory $\{(x_k, y_k)\}_{k=0}^{K}$

Initialisation: $x \leftarrow x_0,\ y \leftarrow y_0,\ k \leftarrow 0$
for $k = 0$ **to** $K_{\max} - 1$ **do**
  compute gradient $\left(\partial f/\partial x, \partial f/\partial y\right)$ at (x, y)
  $(x', y') \leftarrow (x, y) - \eta\,\nabla f(x, y)$
  $\Delta \leftarrow \|(x', y') - (x, y)\|_2$
  if $\Delta < \varepsilon$ **then**
    break
  end if
  $x \leftarrow x',\ y \leftarrow y'$
end for
return (x, y) and trajectory

B. Hyper-Parameters and Stopping Criterion

The following hyper-parameters are specified:

- Step size: $\eta = 0.1$
- Tolerance: $\varepsilon = 10^{-6}$
- Initial point: $(x_0, y_0) = (1, 1)$
- Maximum iterations: $K_{\max} = 1000$

Convergence is declared when $\|(x_{k+1}, y_{k+1}) - (x_k, y_k)\|_2 < \varepsilon$.

IV. Results

Experiments were conducted with a fixed initial guess and iteration budget:

$$(x_0, y_0) = (1, 1), \quad K_{\max} = 1000.$$

The learning rate η and the tolerance ε were systematically varied.

TABLE I
Gradient Descent Hyperparameters and Outcomes

LR	tol	Conv.	$\mathbf{x}^*$	$\mathbf{y}^*$	$\mathbf{f}(\mathbf{x}^*, \mathbf{y}^*)$
0.10	10^{-6}	71	-1.088162	0.544077	-2.280710
0.05	10^{-6}	145	0.605942	-0.302963	1.242733
0.01	10^{-6}	628	0.605924	-0.302916	1.242733
0.01	10^{-3}	201	0.577581	-0.248220	1.245768

The optimisation trajectory obtained with $\eta = 0.10$ and $\varepsilon = 10^{-6}$ (convergence in 71 iterations) is shown in Figure 1.

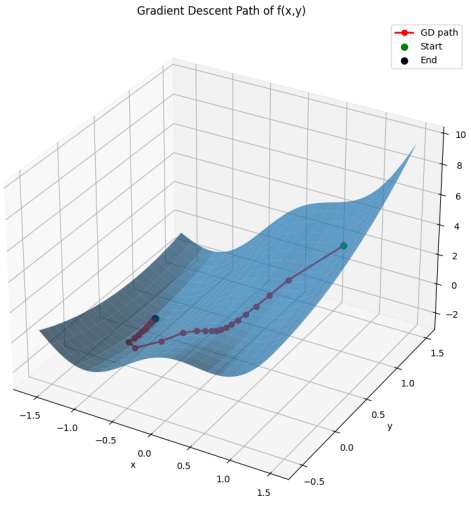

Fig. 1. Gradient descent path on the surface of $f(x, y)$.

V. Discussion

Table I reveals two clear trends: sensitivity to η and dependence on ε.

- **Learning rate η.** $\eta = 0.10$ reaches the global minimiser in 71 iterations. Smaller η (0.05 or 0.01) more than doubles the iterations and becomes trapped in local basins, never reaching $f^* < 0$.
- **Tolerance ε.** For $\eta = 0.01$, relaxing ε from 10^{-6} to 10^{-3} cuts iterations from 628 to 201 but slightly degrades the final objective.

A balance of large η and small ε is required for both speed and precision. Adaptive or momentum-based updates could improve robustness in this non-convex landscape.

VI. Conclusion

Convergence to the global minimum of $f(x, y)$ is achievable using plain gradient descent when η is sufficiently large and ε sufficiently small. A setting of $\eta = 0.10$, $\varepsilon = 10^{-6}$ converged in 71 iterations to $(x^\star, y^\star) = (-1.088162, 0.544077)$ with $f^\star = -2.280710$.