THE CHINESE UNIVERSITY OF HONG KONG
Department of Mathematics
Exercises on Gradient descent method

Gradient Descent Method (GDM) for One-Variable Functions

We now introduce the Gradient Descent Method to obtain an approximate solution to the least-squares problem numerically.

There is a problem of finding the minimum value of a one-variable function $f(x)$ that can be differentiated as follows.

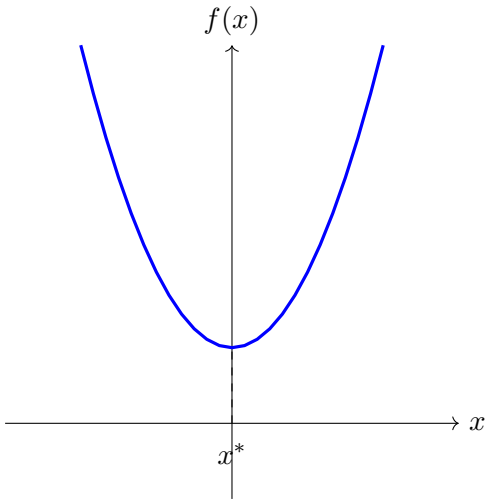

Figure 1: Function $f(x) = x^2 + 1$ and its minimum x^*

Then, the point x^* giving the minimum value of the function (it is called the optimal solution) by Fermat's critical point theorem satisfies the following:

$$f'(x^*) = 0.$$

Therefore, we can determine which of the solutions satisfying the equation $f'(x) = 0$ is the optimal solution. However, when the function is complex, it is not easy to find a critical point by solving the equation. In this case, the critical point is obtained numerically. In this section, we look at the gradient descent method, which is the most popular numerical optimization method to find the minimum value of a given function. The basic idea of the gradient descent method is to find the slope of the function, move it toward downhill, and repeat it until it reaches the extreme value.

Gradient Descent Method Algorithm

- **Step 1** Set an initial iterate x_1, tolerance $0 \leq \epsilon \ll 1$, initial learning rate η (eta), and iteration number $k := 1$.
- **Step 2** Compute $g_k = f'(x_k)$. If $|g_k| \leq \epsilon$, then stop.
- **Step 3** Set $x_{k+1} = x_k - \eta g_k$, $k := k + 1$ and go to **Step 2**.

Let's explain this GDM algorithm. Set x_k and compute $g_k = f'(x_k)$. If $|g_k| < \epsilon$ (it satisfies the critical point theorem within the margin of error we allow), then the algorithm stops and will give x_k as the optimal solution. Here, ϵ (epsilon, tolerance) satisfies $0 \leq \epsilon \ll 1$. A notation

" $\ll$ " in this inequality means that the epsilon is much smaller than 1. Hence, we give a small tolerance epsilon $\epsilon = 10^{-6}$, which is very close to zero, and with a learning rate η. And let the iteration number k be 1. If $|g_k| > \epsilon$, $x_{k+1} = x_k - \eta g_k$ is determined using the formula. Then $g_k = f'(x_k)$ is calculated to determine whether or not the critical point theorem is satisfied. If the critical point theorem is not satisfied, then x_1 is determined similarly. In this way, we create $x_1, x_2, \ldots (\to x^*)$. If the value of $g_k = f'(x_k)$ is within a given tolerance after some repeated steps $x_{k+1} = x_k - \eta f'(x_k)$, then the algorithm stops. We expect some x_k or the limit x^* to satisfy $f'(x^*) = 0$.

Let's see how the GDM works in detail. Suppose the slope of the tangent line to the function f is negative at the approximate solution x_k after the k-th iteration. It is $g_k = f'(x_k) < 0$ as shown in the figure below. This means the function decreases when x_k moves from left to right, so we can expect x^* is located on the right side of x_k. If x_k moves from the left to the right, then x_{k+1} is closer to the optimal value x^*. Therefore, we move from x_k to $x_{k+1} = x_k - \eta g_k$ with $-\eta g_k > 0$ direction.

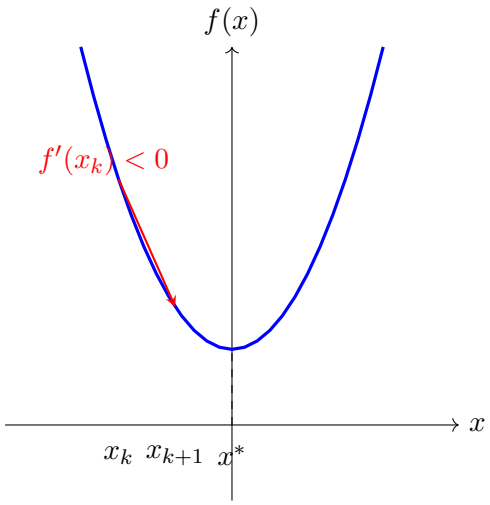

Figure 2: Gradient descent step for $f'(x_k) < 0$

Similarly, the slope of the tangent line to the function f is positive at the approximate solution x_k after the k-th iteration. It is $g_k = f'(x_k) > 0$ as shown in the figure below. It means the function is increasing when x_k moves from left to right, so we can expect x^* is located on the left side of x_k. If x_k moves to the left, then x_{k+1} is closer to the optimal value x^*. Therefore, we move from x_k to $x_{k+1} = x_k - \eta g_k$ with $-\eta g_k < 0$ direction. If we repeat the process until x_k moves to x^*, then $f'(x_k)$ will converge to 0. In this way, we get the approximate solution x_k.

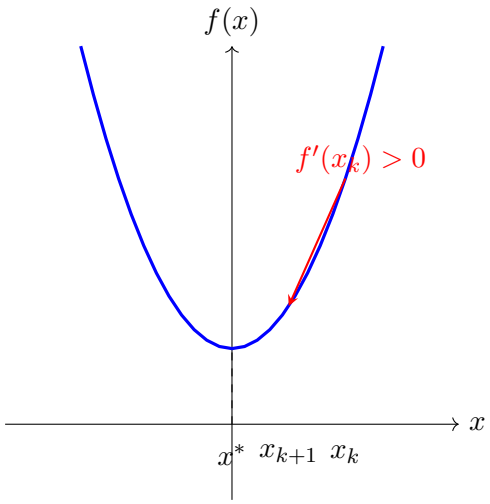

Figure 3: Gradient descent step for $f'(x_k) > 0$

This shows that the GDM will find $x_1, x_2, x_3, \ldots$ that satisfy

$$f(x_1) > f(x_2) > f(x_3) > \ldots.$$

Here, η determines the magnitude of movements. We call this "the learning rate". If the learning rate is too large, x_{k+1} can pass over x^*, or even the function value may increase. So we should be careful to choose a reasonable learning rate η.

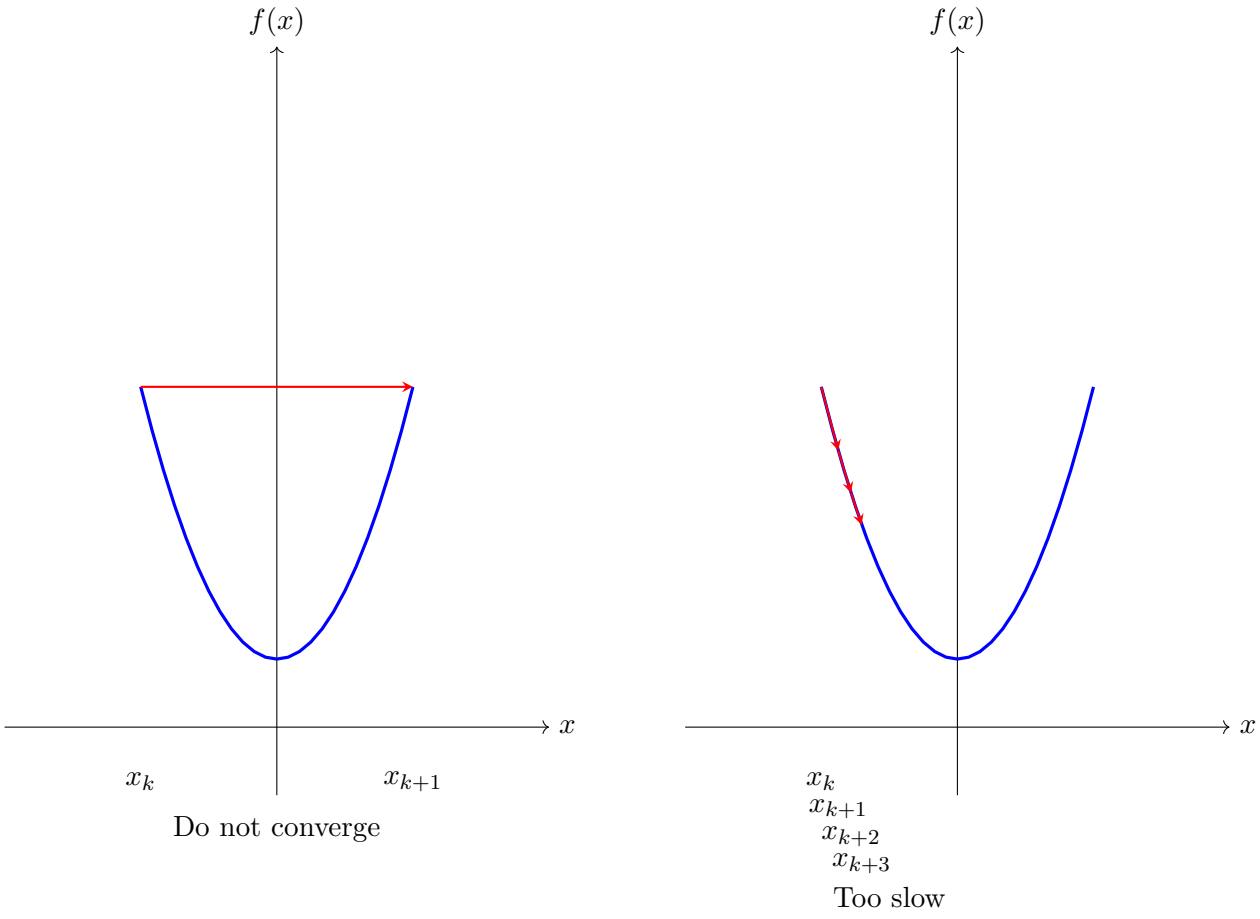

Figure 4: Effect of learning rate: too large (left) vs too small (right)

Parameter	Description
Initial iterate x_1	Starting point for the algorithm
Tolerance ϵ	Stopping threshold for $\lvert f'(x_k)\rvert$ (typically 10^{-6})
Learning rate η	Step size for updates; controls convergence speed/stability
Update rule	$x_{k+1} = x_k - \eta f'(x_k)$
Convergence condition	$\lvert f'(x_k)\rvert \leq \epsilon$

Table 1: Gradient Descent Method (GDM) parameters

Example 1: Gradient Descent for Single-Variable Function

Find the minimum of $f(x) = 2x^2 - 3x + 2$. Take $x_1 = 0$, $\eta = 0.1$, and $\epsilon = 10^{-6}$ in the following code.

Solution

From $f'(x) = 4x - 3 = 0$, the critical point becomes $x = \frac{3}{4} = 0.75$. Since $f(x)$ is convex (opens upward), the minimum value

$$f\left(\frac{3}{4}\right) = \frac{7}{8} = 0.875$$

is obtained at $x = \frac{3}{4}$.

The output of the algorithm is:

- Algorithm Succeed!
- $x^* = 0.749999834194560$
- $|g^*| = 6.63221759289456e - 7$
- $f(x^*) = 0.875000000000055$
- Iteration number = 31

In the above example, the points created by the gradient descent method $(x_1, f(x_1))$, $(x_2, f(x_2))$, $(x_3, f(x_3))$, ... are shown on the coordinate plane along with the graph of the function $y = f(x)$. From the figure, it is easy to see that the sequence converged to the point $(x^*, f(x^*))$ on the curve with the minimum value.

It can be seen that the sequence converges to the optimal solution x^*.

Summary of Example 1 Gradient Descent Results

Parameter	Value
Function	$f(x) = 2x^2 - 3x + 2$
Derivative	$f'(x) = 4x - 3$
Initial iterate x_1	0.0
Tolerance ϵ	10^{-6}
Learning rate η	0.1
Converged solution x^*	0.75
Minimum value $f(x^*)$	0.875
Iterations to converge	31

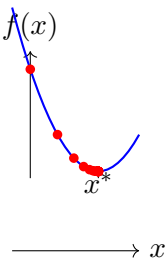

Figure 5: Gradient Descent Path on $f(x) = 2x^2 - 3x + 2$

Example 2: Gradient Descent for Single-Variable Function

Find the minimum value of $f(x) = 9x^2 - 7x + 6$. Use $x_1 = 0$, $\eta = 0.1$, and $\epsilon = 10^{-6}$ with the above GDM code.

Solution First, compute the derivative:

$$f'(x) = 18x - 7.$$

Setting $f'(x) = 0$ gives the critical point:

$$18x - 7 = 0 \implies x^* = \frac{7}{18} \approx 0.3889.$$

Since $f''(x) = 18 > 0$, the function is convex, so $x^* = \frac{7}{18}$ is the minimum point. The minimum value is:

$$f\left(\frac{7}{18}\right) = 9\left(\frac{7}{18}\right)^2 - 7\left(\frac{7}{18}\right) + 6 = \frac{167}{36} \approx 4.6389.$$

Summary of Example 2 Gradient Descent Results

Parameter	Value
Function	$f(x) = 9x^2 - 7x + 6$
Derivative	$f'(x) = 18x - 7$
Initial iterate x_1	0.0
Tolerance ϵ	10^{-6}
Learning rate η	0.1
True minimum x^*	$7/18 \approx 0.3889$
Minimum value $f(x^*)$	$167/36 \approx 4.6389$

Application of the Gradient Descent Method

The GDM introduced in the first session is for minimizing a function $f(x)$ of one variable. Let's generalize the algorithm for minimizing a function of several variables. The least-squares problem we learned is a multivariate function $f(\mathbf{x})$ with at least two independent variables. The least-squares problem was solved using the error function $E(\mathbf{u})$ transformed using multiple variables. The least-squares problem can also be solved using gradient descent method (GDM).

GDM Algorithm for minimizing a multi-variable function

- **Step 1** Set an initial iterate $\mathbf{u}_1 \in \mathbb{R}^n$, tolerance $0 \leq \epsilon < 1$, initial learning rate η (eta), and iteration number $k := 1$.
- **Step 2** Compute $\mathbf{g}_k = \nabla E(\mathbf{u}_k)$. If $\|\mathbf{g}_k\| \leq \epsilon$, then stop.

- **Step 3** Set $\mathbf{u}_{k+1} = \mathbf{u}_k - \eta \mathbf{g}_k$, $k := k + 1$ and go to **Step 2**.

All the steps of the GDM algorithm for minimizing a multi-variable function are the same as for a one-variable function.

Correspondence between One-Variable and Multi-Variable Functions

One variable function	Multi-variable function
Scalar x	Vector $\mathbf{u}$
Absolute value $\|a\|$	Norm of vector $\|\mathbf{g}\|$
Derivative $f'(x)$	Gradient $\nabla E(\mathbf{u})$

Let x, y be independent variables and z be a third variable. Now we can consider a function $z = f(x, y)$, which is dependent on x and y. Functions of several variables can be defined in the same manner.

In a 3-dimensional (coordinate) space, $z = f(x, y)$ can be considered as a point (x, y, z). As x and y move, the graph of $z = f(x, y)$ becomes a surface. For example, the graph of a two-variable function $z = f(x, y) = -xye^{-x^2-y^2}$ can be drawn using the codes as follows.

The graph of f in the figure above intuitively shows that f has a local maximum value at the peak and a local minimum value at the valley. To find those critical points, the notion of a *gradient of a multi-variable function* is required.

The gradient of a function of two variables $f(x, y)$ is defined as follows:

$$\operatorname{grad} f(x, y) = \nabla f(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}.$$

The gradient of $f(x, y)$ is a 2×1 vector, where $\frac{\partial f}{\partial x}$ means the partial derivative of f with respect to x. The partial derivative with respect to x considers other variables except x is constant. Similarly, $\frac{\partial f}{\partial y}$ means the partial derivative of f with respect to y.

The gradient of $z = f(x, y) = -xye^{-x^2-y^2}$ can be found as follows.

Solution

The computed gradient result is:

$$(2x^2ye^{-x^2-y^2} - ye^{-x^2-y^2},\ 2xy^2e^{-x^2-y^2} - xe^{-x^2-y^2})$$

$$\operatorname{grad} f(x, y) = \nabla f(x, y) = \left(2x^2ye^{-(x^2+y^2)} - ye^{-(x^2+y^2)},\ 2xy^2e^{-(x^2+y^2)} - xe^{-(x^2+y^2)}\right)$$

Show Calculations

The gradient of a two-variable scalar function $f(x, y)$ is defined as the vector of its first-order partial derivatives:

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x},\ \frac{\partial f}{\partial y}\right)$$

where ∇ denotes the nabla operator, and $\operatorname{grad} f$ is the equivalent gradient notation.

We compute the partial derivatives for $f(x,y) = -xye^{-x^2-y^2}$ using the **product rule** for differentiation:

$$\frac{\partial}{\partial x}(u \cdot v) = \frac{\partial u}{\partial x} \cdot v + u \cdot \frac{\partial v}{\partial x}, \quad \frac{\partial}{\partial y}(u \cdot v) = \frac{\partial u}{\partial y} \cdot v + u \cdot \frac{\partial v}{\partial y}.$$

1: Compute the partial derivative with respect to x

Let $u = -xy$ and $v = e^{-x^2-y^2}$. Then:

$$\begin{aligned}\frac{\partial f}{\partial x} &= \frac{\partial}{\partial x}(-xy) \cdot e^{-x^2-y^2} + (-xy) \cdot \frac{\partial}{\partial x}\left(e^{-x^2-y^2}\right) \\ &= -ye^{-x^2-y^2} + (-xy) \cdot e^{-x^2-y^2} \cdot (-2x) \\ &= 2x^2ye^{-(x^2+y^2)} - ye^{-(x^2+y^2)}.\end{aligned}$$

2: Compute the partial derivative with respect to y

Let $u = -xy$ and $v = e^{-x^2-y^2}$. Then:

$$\begin{aligned}\frac{\partial f}{\partial y} &= \frac{\partial}{\partial y}(-xy) \cdot e^{-x^2-y^2} + (-xy) \cdot \frac{\partial}{\partial y}\left(e^{-x^2-y^2}\right) \\ &= -xe^{-x^2-y^2} + (-xy) \cdot e^{-x^2-y^2} \cdot (-2y) \\ &= 2xy^2e^{-(x^2+y^2)} - xe^{-(x^2+y^2)}.\end{aligned}$$

Gradient of Multi-Variable Functions Similarly, the gradient of a multi-variable function with three or more variables f can be obtained. In general, the gradient of the n-variable function $f(x_1, x_2, \ldots, x_n)$ is defined as:

$$\operatorname{grad} f(x_1, x_2, \ldots, x_n) = \nabla f(x_1, x_2, \ldots, x_n) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}.$$

Example 3: Least-Squares Problem with Gradient Descent

Given:

x_i	u_i
0	1
1	3
2	4
3	4

The first example of the least-squares problem was as follows. For each data (x_i, u_i), let $\hat{u}_i$ be the value obtained by substituting x_i in a linear function $u = a + bx$. Thus:

$$\hat{u}_i = a + bx_i.$$

If this equation's solution does not exist, it is possible to find a, b where the square of error $(u_i - \hat{u}_i)^2$ is minimized. The error function $E(\mathbf{u})$, which is the sum of squared errors $(u_i - \hat{u}_i)^2$,

is defined as follows (the factor $\frac{1}{2}$ is included for computational convenience and does not affect the conclusion):

$$\begin{aligned} E(\mathbf{u}) = E(a,b) &= \frac{1}{2}\left\{(a-1)^2 + (a+b-3)^2 + (a+2b-4)^2 + (a+3b-4)^2\right\} \\ &= \frac{1}{2}\|A\mathbf{u} - \mathbf{y}\|^2. \end{aligned}$$

1. Matrix Formulation The linear model $u = a + bx$ gives the design matrix A, parameter vector $\mathbf{u}$, and target vector $\mathbf{y}$:

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} a \\ b \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 4 \end{bmatrix}.$$

The error function is:

$$E(\mathbf{u}) = \frac{1}{2}\|A\mathbf{u} - \mathbf{y}\|^2.$$

2. Gradient of the Error Function Using vector calculus, the gradient is:

$$\nabla E(\mathbf{u}) = A^T(A\mathbf{u} - \mathbf{y}).$$

Expanded component-wise:

$$\begin{aligned} \frac{\partial E}{\partial a} &= (a-1) + (a+b-3) + (a+2b-4) + (a+3b-4), \\ \frac{\partial E}{\partial b} &= 0\cdot(a-1) + 1\cdot(a+b-3) + 2\cdot(a+2b-4) + 3\cdot(a+3b-4). \end{aligned}$$

3. GDM Update Rule Iteratively update the parameter vector:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \eta\,\nabla E(\mathbf{u}_k).$$

Stop when $\|\nabla E(\mathbf{u}_k)\| < \epsilon$.

Employ `Python` to use the GDM to find the approximate solution $\hat{\mathbf{u}}$ that minimizes $E(\mathbf{u})$.

Set an initial iterate $\mathbf{u}_1 = (-2.5, -2.5)$, tolerance $\epsilon = 10^{-6}$, and initial learning rate $\eta = 0.1$.

Summary of GDM least-squares linear fit

Parameter	Value
Initial iterate $\mathbf{u}_1$	$(-2.5, -2.5)$
Tolerance ϵ	10^{-6}
Learning rate η	0.1
Converged solution $\mathbf{u}^*$	$(1.5, 1.0)$
Minimum error $E(\mathbf{x}^*)$	0.5
Iterations to converge	118
Least-squares line	$y = x + \frac{3}{2}$

Therefore, the line we get from the least-squares solution is

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

Example 4: Least-Squares Problem with Gradient Descent

Employ `Python` to use the GDM and the same data above to find a quadratic function $y = a + bx + cx^2$ that best fits the given data.

We employ the Gradient Descent Method (GDM) to find the quadratic function $y = a+bx+cx^2$ that best fits the given data. The error function $E(\mathbf{u})$ is defined as:

$$\begin{aligned} E(a,b,c) &= \frac{1}{2}\left\{(a-1)^2 + (a+b+c-3)^2 + (a+2b+4c-4)^2 + (a+3b+9c-4)^2\right\} \\ &= \frac{1}{2}\|A\mathbf{u}-\mathbf{y}\|^2, \end{aligned}$$

where $\mathbf{u} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ is the parameter vector.

1. Matrix Form Construction The quadratic model is $y = a + bx + cx^2$. From the error function, the data points are: $(x, y) = (0, 1)$, $(1, 3)$, $(2, 4)$, $(3, 4)$.

The design matrix A, parameter vector $\mathbf{u}$, and target vector $\mathbf{y}$ are:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 4 \end{bmatrix}.$$

The error function is the mean squared error (scaled by $1/2$):

$$E(\mathbf{u}) = \frac{1}{2}\|A\mathbf{u}-\mathbf{y}\|^2.$$

2. Gradient of the Error Function (Critical for GDM) The gradient $\nabla E(\mathbf{u})$ for GDM update is:

$$\nabla E(\mathbf{u}) = A^T(A\mathbf{u}-\mathbf{y}).$$

Expand the gradient components for (a, b, c):

$$\nabla E = \begin{bmatrix} \frac{\partial E}{\partial a} \\ \frac{\partial E}{\partial b} \\ \frac{\partial E}{\partial c} \end{bmatrix} = A^T(A\mathbf{u}-\mathbf{y}).$$

3. GDM Update Rule Iteratively update parameters with learning rate η:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \eta \cdot \nabla E(\mathbf{u}_k).$$

Result

The GDM converges to the optimal least-squares parameters:

$$a \approx 1.05, \quad b \approx 1.45, \quad c \approx -0.25.$$

The best-fit quadratic function is:

$$y = 1.05 + 1.45x - 0.25x^2.$$