Applying Gradient Descent

No closed-form solution like linear regression

Solution:

Iterative Optimization: Gradient Descent

Update Rule:

$$w \leftarrow w - \eta \nabla L(w)$$

Gradient:

$$\partial L/\partial w = (\hat{y} - y)x$$

(similar form to linear regression!)

Learning Rate η:

Controls step size (typically 0.001-0.1)

Variants:

- Batch Gradient Descent
- Mini-batch Gradient Descent
- Stochastic Gradient Descent

Convergence

Monitor on validation set:

Loss

Accuracy

Iterative Process

Initialize → Compute Gradient → Update ↻

Step-by-Step: Gradient Descent in Action

1

Initialize Parameters

Movement in Loss Space

Start with initial weights (often random or zeros)

$$w_0 = 0.5$$

2

Compute Prediction

Forward pass through the model

$$\hat{y} = \sigma(w_0 x) = \sigma(0.5 \times 2.0) = 0.73$$

3

Calculate Loss

Measure prediction error

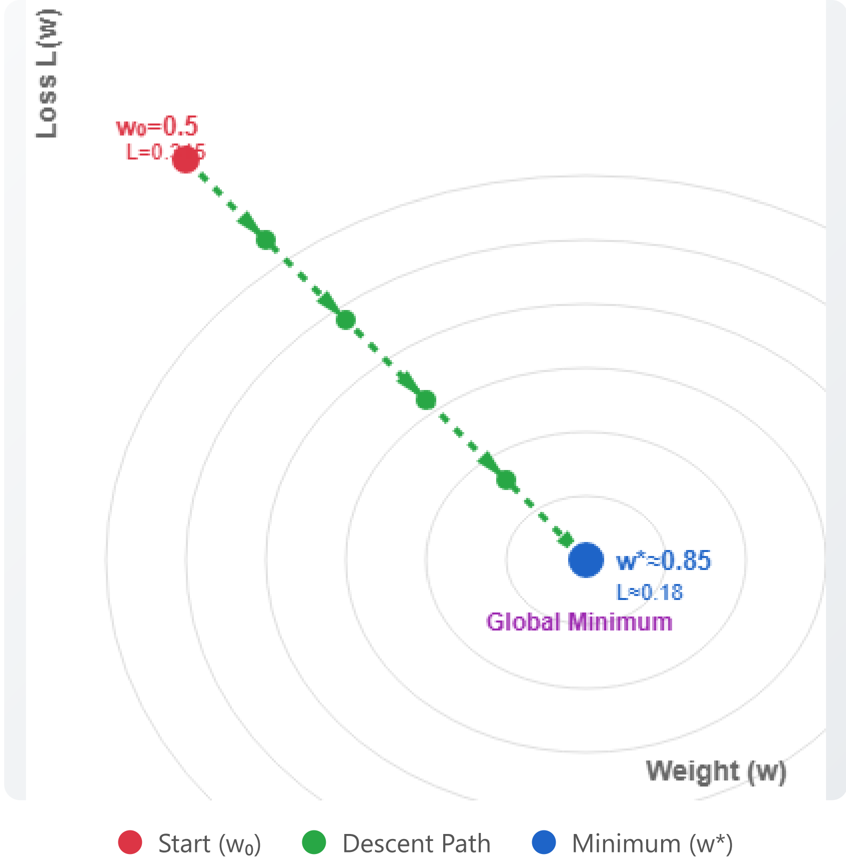
$$L = -[y \log(\hat{y}) + (1-y) \log(1-\hat{y})]$$
$$L = 0.315$$

4

Compute Gradient

Calculate direction of steepest ascent

$$\partial L/\partial w = (\hat{y} - y)x = (0.73 - 1) \times 2.0$$
$$\nabla L = -0.54$$



5

Update Parameters

Move in opposite direction of gradient

$w_1 = w_0 - \eta \nabla L$

$w_1 = 0.5 - 0.1 \times (-0.54) = 0.554$

6

Repeat

Continue until convergence

Iterate steps 2-5 until $\Delta L < \varepsilon$

Concrete Example

Training Data:
x = 2.0, y = 1 (positive class)

Learning Rate:
$\eta = 0.1$

Initial Weight:
$w_0 = 0.5$

After Update:
$w_1 = 0.554$ (↓ Loss: 0.315 → 0.289)