Phase 1 — Neural Network Math (SGD + BCE, Matches Code)

Goal

Understand the mathematical foundations of neural networks so they can be implemented *from scratch in C++*, without machine learning libraries.

- Time: approximately 4–6 days
- Difficulty: Medium

Notation (Important)

We explicitly distinguish between the true label and the model prediction:

- y denotes the **true label** provided by the dataset (binary: $y \in \{0, 1\}$)
- $\hat{y}$ denotes the **model prediction** (a probability in $(0, 1)$)
- z denotes a **pre-activation** value ($z = wx + b$)
- a denotes an **activation** value ($a = \phi(z)$)

Throughout this document, for binary classification:

$$a_2 = \hat{y}.$$

Step 2 — Forward Pass (Two Layers, SGD Case)

We train on one example at a time: $(\mathbf{x}, y)$.

Let $\mathbf{x} \in \mathbb{R}^n$. Choose a hidden width h.

Hidden Layer (ReLU)

$$\mathbf{z}_1 = W_1\mathbf{x} + \mathbf{b}_1, \qquad \mathbf{a}_1 = \mathrm{ReLU}(\mathbf{z}_1),$$

where

$$W_1 \in \mathbb{R}^{h\times n}, \quad \mathbf{b}_1 \in \mathbb{R}^h, \quad \mathrm{ReLU}(t) = \max(0, t) \text{ (applied elementwise).}$$

Output Layer (Sigmoid)

For BCE we require a probability output, so we use sigmoid at the final layer:

$$z_2 = W_2\mathbf{a}_1 + b_2, \qquad \hat{y} = a_2 = \sigma(z_2),$$

where

$$W_2 \in \mathbb{R}^{1\times h}, \quad b_2 \in \mathbb{R}, \quad \sigma(t) = \frac{1}{1 + e^{-t}}.$$

Step 3 — Loss (Binary Cross-Entropy, Full Form)

For binary classification ($y \in \{0, 1\}$) the binary cross-entropy loss is:

$$\boxed{L(\hat{y}, y) = -\left[y \ln(\hat{y}) + (1-y)\ln(1-\hat{y})\right]}$$

This single formula covers both cases:

$$y = 1 \Rightarrow L = -\ln(\hat{y}), \qquad y = 0 \Rightarrow L = -\ln(1-\hat{y}).$$

Step 4 — Backpropagation (SGD: One Example)

We compute gradients for this one example, then update immediately.

Define Error Signals (Deltas)

Define:

$$\delta_2 = \frac{\partial L}{\partial z_2}, \qquad \boldsymbol{\delta}_1 = \frac{\partial L}{\partial \mathbf{z}_1}.$$

Where Does δ_2 Come From? (Sigmoid + BCE)

We have:

$$\hat{y} = \sigma(z_2), \qquad L(\hat{y}, y) = -\left[y \ln(\hat{y}) + (1-y)\ln(1-\hat{y})\right].$$

Using the chain rule:

$$\frac{\partial L}{\partial z_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2}.$$

Compute each piece:

$$\frac{\partial L}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}, \qquad \frac{\partial \hat{y}}{\partial z_2} = \hat{y}(1-\hat{y}).$$

Multiply:

$$\delta_2 = \left(-\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}\right)\hat{y}(1-\hat{y}) = \hat{y} - y.$$

So in code-matching form:

$$\boxed{\delta_2 = \hat{y} - y.}$$

Output Layer Gradients

Because $z_2 = W_2\mathbf{a}_1 + b_2$:

$$\boxed{\frac{\partial L}{\partial W_2} = \delta_2\, \mathbf{a}_1^T, \qquad \frac{\partial L}{\partial b_2} = \delta_2.}$$

Entry-wise (what loops compute):

$$\left(\frac{\partial L}{\partial W_2}\right)_{1j} = \delta_2 (a_1)_j.$$

Backpropagate Into the Hidden Layer

First move the error back through W_2:

$$\frac{\partial L}{\partial \mathbf{a}_1} = W_2^T \delta_2.$$

ReLU derivative (elementwise):

$$\mathrm{ReLU}'(t) = \begin{cases} 1, & t > 0, \\ 0, & t \le 0. \end{cases}$$

Apply the ReLU gate:

$$\boxed{\boldsymbol{\delta}_1 = \left(W_2^T \delta_2\right) \odot \mathrm{ReLU}'(\mathbf{z}_1)}$$

where $\odot$ is elementwise multiplication.

Hidden Layer Gradients

Because $\mathbf{z}_1 = W_1\mathbf{x} + \mathbf{b}_1$:

$$\boxed{\frac{\partial L}{\partial W_1} = \boldsymbol{\delta}_1 \, \mathbf{x}^T, \qquad \frac{\partial L}{\partial \mathbf{b}_1} = \boldsymbol{\delta}_1.}$$

Entry-wise:

$$\left(\frac{\partial L}{\partial W_1}\right)_{ij} = (\delta_1)_i x_j.$$

Step 5 — SGD Parameter Update (Non-Batch)

SGD updates **immediately after one example**.

For each parameter:

$$\boxed{\mathrm{param} \leftarrow \mathrm{param} - \eta \frac{\partial L}{\partial \mathrm{param}}}$$

So:

$$W_2 \leftarrow W_2 - \eta \frac{\partial L}{\partial W_2}, \qquad b_2 \leftarrow b_2 - \eta \frac{\partial L}{\partial b_2},$$
$$W_1 \leftarrow W_1 - \eta \frac{\partial L}{\partial W_1}, \qquad \mathbf{b}_1 \leftarrow \mathbf{b}_1 - \eta \frac{\partial L}{\partial \mathbf{b}_1}.$$

Order of Operations (Exactly What SGD Code Does)

For one training example $(\mathbf{x}, y)$:

1. Forward: compute $\mathbf{z}_1, \mathbf{a}_1, z_2, \hat{y}$.
2. Compute loss $L(\hat{y}, y)$ (BCE).
3. Compute $\delta_2 = \hat{y} - y$.
4. Compute $\frac{\partial L}{\partial W_2}$ and $\frac{\partial L}{\partial b_2}$.

5. Compute $\boldsymbol{\delta}_1 = (W_2^T \delta_2) \odot \mathrm{ReLU}'(\mathbf{z}_1)$.
6. Compute $\frac{\partial L}{\partial W_1}$ and $\frac{\partial L}{\partial \mathbf{b}_1}$.
7. Update $W_1, \mathbf{b}_1, W_2, b_2$ immediately (SGD).