Mathematical Foundations of an L-layer Neural Network for MNIST Digit Recognition

1 Introduction

We implement a fully connected deep neural network to classify handwritten digits from the MNIST dataset. The architecture follows the structure

$$[\text{Linear} \to \text{ReLU}]^{L-1} \to \text{Linear} \to \text{Softmax}$$

with cross-entropy loss and optimization via Adam.

2 Parameter Initialization

For layer l, we define:

- Weights $W^{[l]} \in \mathbb{R}^{n^{[l]} \times n^{[l-1]}}$,
- Biases $b^{[l]} \in \mathbb{R}^{n^{[l]} \times 1}$.

He initialization is used to preserve variance across ReLU layers:

$$W^{[l]} \sim \mathcal{N}\left(0,\ \frac{2}{n^{[l-1]}}\right)$$

3 Forward Propagation

3.1 Linear Transformation

For input $A^{[l-1]}$, the pre-activation is

$$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$$

3.2 Activation Functions

ReLU:

$$A^{[l]} = \max(0, Z^{[l]})$$

Softmax (output layer): For class k and sample i,

$$A^{[L]}_{k,i} = \frac{e^{Z^{[L]}_{k,i}}}{\sum_{j=1}^{C} e^{Z^{[L]}_{j,i}}}$$

4 Loss Function

We minimize the categorical cross-entropy loss:

$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{C} Y_{k,i}\log A_{k,i}^{[L]}$$

where m is the batch size and C the number of classes.

5 Backward Propagation

5.1 Softmax + Cross-Entropy

Combining softmax with cross-entropy yields a simplified gradient:

$$\frac{\partial \mathcal{L}}{\partial Z^{[L]}} = A^{[L]} - Y$$

5.2 Linear Backward

From $Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$, we derive

$$\begin{aligned} dW^{[l]} &= \frac{1}{m} dZ^{[l]} (A^{[l-1]})^T \\ db^{[l]} &= \frac{1}{m}\sum_{i=1}^{m} dZ^{[l](i)} \\ dA^{[l-1]} &= (W^{[l]})^T dZ^{[l]} \end{aligned}$$

5.3 ReLU Backward

The gradient of ReLU is

$$\frac{\partial A^{[l]}}{\partial Z^{[l]}} = \begin{cases} 1 & Z^{[l]} > 0, \\ 0 & \text{otherwise}, \end{cases}$$

so

$$dZ^{[l]} = dA^{[l]} \odot \mathbf{1}_{Z^{[l]}>0}$$

6 Optimization Algorithms

6.1 Gradient Descent

The parameters are updated as

$$W^{[l]} \leftarrow W^{[l]} - \alpha dW^{[l]}, \quad b^{[l]} \leftarrow b^{[l]} - \alpha db^{[l]},$$

where α is the learning rate.

6.2 Adam Optimizer

For gradient g_t at step t, Adam computes:

$$v_t = \beta_1 v_{t-1} + (1-\beta_1) g_t$$
$$s_t = \beta_2 s_{t-1} + (1-\beta_2) g_t^2$$

Bias correction:

$$\hat{v}_t = \frac{v_t}{1-\beta_1^t}, \quad \hat{s}_t = \frac{s_t}{1-\beta_2^t}$$

Parameter update:

$$\theta \leftarrow \theta - \alpha \frac{\hat{v}_t}{\sqrt{\hat{s}_t} + \epsilon}$$

7 Mini-Batch Training

Training uses mini-batches of size m_b to estimate the gradient, which reduces variance and improves efficiency.

8 Prediction and Evaluation

Predicted class for sample i is

$$\hat{y}^{(i)} = \arg\max_k A_{k,i}^{[L]}$$

The accuracy is

$$\text{Acc} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}(\hat{y}^{(i)} = y^{(i)})$$