Practice Deep Learning Lab 1

Name: Shrushty Dhamange
PRN: 202201070016
Batch:T2

==========

Neural Network from Scratch

Task:

1. **Output Layer:**
   - Size: 10 neurons (one for each digit class, 0–9).
   - Activation Function: Softmax is used for probabilistic classification.
   - Each output neuron represents the predicted probability of the corresponding digit.

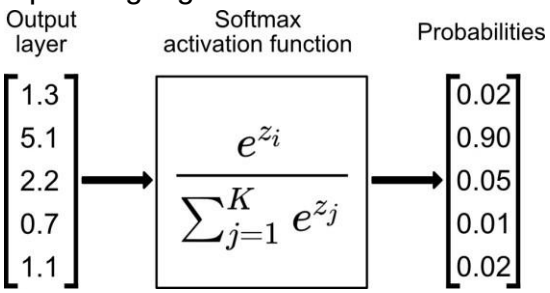


Diagram by Towards Data Science

Forward Pass

The forward pass computes predictions by propagating input data through the network:

1. **Input to Hidden Layer**:
   - Compute $Z1=W1.X+b1$, where W1 is the weight matrix, b1 is the bias, and X is the input.
   - Apply ReLU activation: $A1=ReLU(Z1)$.
2. **Hidden to Output Layer**:
   - Compute $Z2=W2.A1+b2$, where W2and b2 are weights and biases for the second layer.
   - Apply softmax activation: $A2=softmax(Z2)$.
3. The result, A2, represents the probabilities for each class. The class with the highest probability is the predicted label.

Backpropagation

Backpropagation is used to compute gradients and adjust weights to minimize the loss function:

1. **Error at the Output Layer**:
    - Calculate the difference between the predicted probabilities (A2A_2A2) and the true labels (YYY).

2. **Gradients for Output Layer**:
    - Compute dZ2=A2−Y
    - Calculate weight and bias gradients:

$$dW_2 = \frac{1}{m} dZ_2 \cdot A_1^T, \quad db_2 = \frac{1}{m} \sum dZ_2$$

3. **Gradients for Hidden Layer**:
    - Backpropagate the error to the hidden layer:

$$dZ_1 = W_2^T \cdot dZ_2 \cdot \text{ReLU\_deriv}(Z_1)$$

    - Compute weight and bias gradients:

$$dW_1 = \frac{1}{m} dZ_1 \cdot X^T, \quad db_1 = \frac{1}{m} \sum dZ_1$$

4. **Weight Update**:
    - Update weights and biases using gradient descent:

$$W_1 = W_1 - \alpha dW_1, \quad b_1 = b_1 - \alpha db_1$$

$$W_2 = W_2 - \alpha dW_2, \quad b_2 = b_2 - \alpha db_2$$

Loss Function

The **Cross-Entropy Loss** is used for classification tasks:

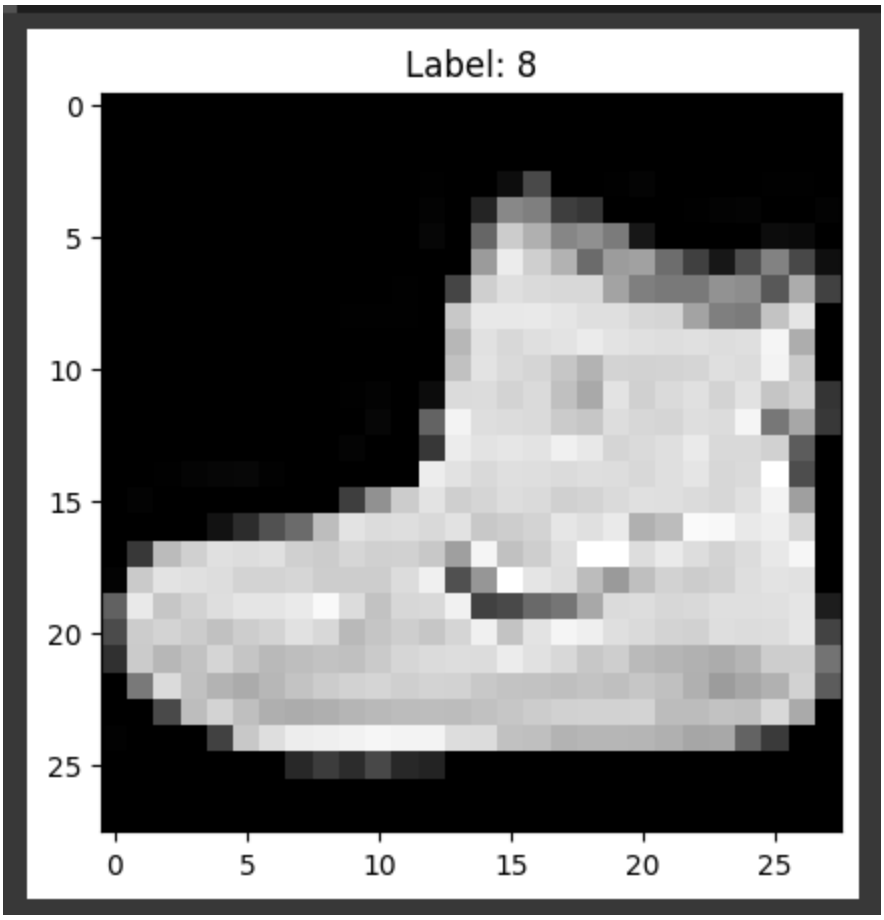
$$L = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{10} Y_{ij}\log(A_{ij})$$

- Yij: Actual label (one-hot encoded).
- Aij: Predicted probability for class j.
  This loss measures how far the predicted probabilities are from the true labels.

Optimization

Gradient Descent is used to minimize the loss function:

1. Compute gradients of the loss with respect to weights and biases (as described in backpropagation).
2. Update parameters (weights and biases) iteratively using the learning rate α\alphaα.



Declaration:

GitHub Repository Link: https://github.com/215shrushty/neural-network-from-scratch.git

Code File:

https://colab.research.google.com/drive/1-ytLMKfLPPj8X-9hsw8Mug-m1WgyUz1r#scrollTo=amUMzOvxROut

https://colab.research.google.com/drive/13ITRhZRWrZK2FSpls0qU09hvBNSLy0-h#scrollTo=waE9gA7hG-VZ