Assignment 1 – Report

Training Artificial Neural Networks Using Cross-Entropy Loss

Objective

This report derives the formulas for updating weights in a feedforward neural network using the **cross-entropy loss function**. We focus on calculating:

- The error signal δ for each neuron
- The weight update Δw for both output and hidden layer neurons

1. Neural Network Structure

Let:

- x_i - input to neuron j
- w_{ji} - weight from neuron i to neuron j
- b_j - bias of neuron j
- $v_j = \sum_i \quad w_{ji}x_i + b_j$ - net input to neuron j
- $y_j = \phi(v_j)$ - output of neuron j, where ϕ is the activation function

2. Cross-Entropy Loss Function

For binary classification with sigmoid activation:

$$\mathcal{L} = -[d \cdot \log(y) + (1 - d) \cdot \log(1 - y)]$$

Where:

- d - desired output (label)
- y - predicted output

For multi-class classification with softmax activation:

$$\mathcal{L} = -\sum_j \quad d_j \cdot \log(y_j)$$

3. Output Layer Derivation

Activation Function: Sigmoid

$$y_j = \phi(v_j) = \frac{1}{1 + e^{-v_j}}$$

Derivative of Sigmoid

$$\phi'(v_j) = y_j(1 - y_j)$$

Error Signal for Output Neuron

For cross-entropy loss with sigmoid activation, the error simplifies to:

$$\delta_j = y_j - d_j$$

Weight Update Rule

Using gradient descent:

$$\Delta w_{ji} = -\mu \cdot \frac{\partial \mathcal{L}}{\partial w_{ji}} = -\mu \cdot \delta_j \cdot x_i$$

So, the updated weight becomes:

$$w_{ji}^{\text{new}} = w_{ji}^{\text{old}} - \mu \cdot \delta_j \cdot x_i$$

4. Hidden Layer Derivation

Error Signal for Hidden Neuron

Hidden neurons receive error signals from the next layer:

$$\delta_j = \phi'(v_j) \cdot \sum_k \delta_k \cdot w_{kj}$$

Where:

- δ_k - error from neuron k in the next layer
- w_{kj} - weight from hidden neuron j to output neuron k

Weight Update Rule

$$\Delta w_{ji} = -\mu \cdot \delta_j \cdot x_i$$

5. Training Procedure Summary

1. **Initialize weights** w_{ji}randomly
2. **Forward pass**: compute outputs y_j
3. **Compute loss** using cross-entropy
4. **Backpropagate errors**:
    - Output layer: $\delta_j = y_j - d_j$
    - Hidden layer: $\delta_j = \phi'(v_j) \cdot \sum_k \quad \delta_k \cdot w_{kj}$
5. **Update weights**:
    - $w_{ji} \leftarrow w_{ji} - \mu \cdot \delta_j \cdot x_i$
6. **Repeat** for all training samples until convergence

6. Example: One Output Neuron

Given:

- Input: $x = [x_1, x_2]$
- Weights: $w = [w_1, w_2]$
- Bias: b
- Desired output: d

Steps:

1. Compute net input: $v = w_1 x_1 + w_2 x_2 + b$
2. Apply activation: $y = \frac{1}{1+e^{-v}}$
3. Compute error signal: $\delta = y - d$
4. Update weights: $\Delta w_1 = -\mu \cdot \delta \cdot x_1 \Delta w_2 = -\mu \cdot \delta \cdot x_2$

Report Location

https://github.com/sharaba22/cda01/tree/main/Assignment%201