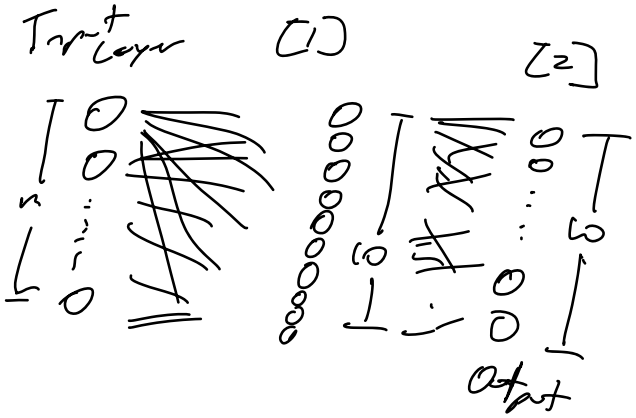
Neural Network From Scratch
Math + Logic

Training + Testing Images

Each Image: 28 × 28 = 784 px ⟹ Each Pixel has value (0, 255)

Let $X =$ [x_1, x_2, …, x_m]^T = [x_1 x_2 … x_m] (784 × m)



Forward Propagation

$A^{[0]} = X$ // Input Layer

$Z^{[1]} = W^{[1]} \cdot A^{[0]} + b^{[1]}$

(10 × 784)(784 × m) + (10 × m)

↳ 10 × m

// weights · matrix + bias

$A^{[1]} = g(Z^{[1]})$ (Leaky) $= ReLU(Z^{[1]})$

// Activation Function

↳ Steps from Linear Combination

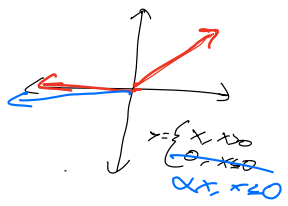
$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$ Only

(10 × m) (10 × 10)(10 × m) + (10 × m)

$A^{[2]} = g_2(Z^{[2]}) = softmax(Z^{[2]})$

↳ Activation for Output Layer

Leaky ReLU



Softmax:

Output → SM → Probabilities

10 [ ] $\Rightarrow \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \Rightarrow$ [0.02, 0.00, …, P]

Backward Propagation

Error: $dZ^{[2]} = A^{[2]} - Y$ (10×m, 10×m, 10×m)

To Find W, b contribution to Error

$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$

$db^{[2]} = \frac{1}{m} \sum dZ^{[1]}$ // Avg of total Error (10×1, 10×1)

Layer 2

$dZ^{[1]}_{10\times m} = W^{[2]T} dZ^{[2]}$

$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$

$db^{[1]} = \frac{1}{m} \sum dZ^{[1]}$

Layer 1

Update Params

Let α be a learning rate

$W^{[1]} := W^{[1]} - \alpha dW^{[1]}$

$b^{[1]} := b^{[1]} - \alpha db^{[1]}$

$W^{[2]} := W^{[2]} - \alpha dW^{[2]}$

$b^{[2]} := b^{[2]} - \alpha db^{[2]}$

Reconfiguring Weights

Update: Use Adam paper to tune dynamically Change

Model