Neural Network Notes

The Problem: Digit Classification (MNIST Database)

784 ⟹ output what digit that image represents: 0, 1, 2 ... 9 (10 classes)

28x28 pixel training images

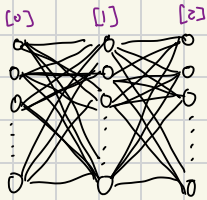
m training images

each row is an example:

$X = \begin{bmatrix} — x^{(1)} — \\ — x^{(2)} — \\ \vdots \\ — x^{(m)} — \end{bmatrix}^T \rightarrow \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$ ↘ 784 rows per pixel

2-Layer Neural Network

↳ each of 784 pixels maps to a Node



0 layer -> just input

1st layer -> 1st hidden layer

2nd layer -> output layer

1. Forward Propogation

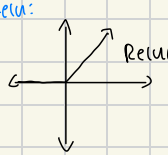
↳ getting output by passing image through network

$A^{[0]} = X$

$Z^{[1]} = W^{[1]} A^{[0]} + b^{[1]}$ → unactivated first layer (weights and bias)
(10x784 784xm)

$A^{[1]} = g(Z^{[1]}) = ReLU(Z^{[1]})$ → activation layer to make it a hidden layer (Relu act. func.)

Relu:

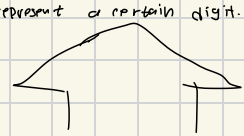


$Relu(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$

$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$ → unactivated 2nd layer

$A^{[2]} = Softmax(Z^{[2]})$ → output-layer → softmax activation fcn $\frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$ → probabilities ↓ $0 \leq x \leq 1$

* The probabilities represent the likeliness that a pixel could represent a certain digit.



2. Backward Propagation

↳ going backward and comparing result w/ actual value

Error of output: how much output is off by

$$dz^{[2]} = A^{[2]} - Y$$

($dz^{[2]}$: loss; $A^{[2]}$: result; Y: acc. value → will one-hot encode i.e. $y=1$: [0, 1, 0, 0, 0, 0, 0, 0, 0, 0])

How much weight/bias contributed to error in output layer:

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

$$db^{[2]} = \frac{1}{m} \sum dz^{[2]} \rightarrow \text{average of absolute error}$$

Error of hidden layer 1: how much hidden layer 1 was off by

$$dz^{[1]} = W^{[2]T} dz^{[2]} \cdot g'(z) \rightarrow$$

- $W^{[2]T} dz^{[2]}$: Applying weights to error from 2nd layer
- $g'(z)$: derivative of Activation func (Undoing to get right error)

How much weight/bias contributes to error in first layer:

$$dW^{[1]} = \frac{1}{m} dz^{[1]}$$

3. Update parameters

$$W^{[1]} = W^{[1]} - \alpha dW^{[1]}$$
$$b^{[1]} = b^{[1]} - \alpha db^{[1]}$$
$$W^{[2]} = W^{[2]} - \alpha dW^{[2]}$$
$$b^{[2]} = b^{[2]} - \alpha db^{[2]}$$

α: learning rate → hyperparameter, not trained by model