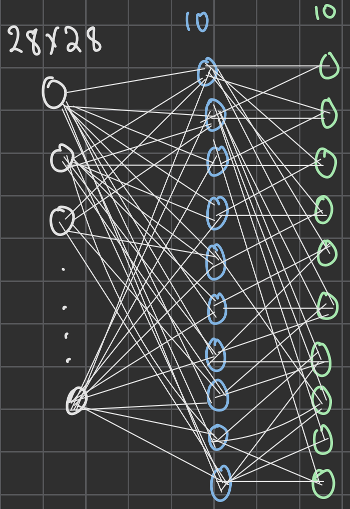
# Image classification software

Digit recognition.
No convolution

this project will be created using a simple neural network, without using neural network libraries.

Input: 28X28 pixel Image of a number.
Output: 10 neurons, each of which describes the probability of the given image being that number from 0 to 10.
* By the end of the training we aim to have one neuron in the final layer significantly light up.



each layer is connected to the layer before it by the weighted sum of the previous layer. more on the math in the next page.

# Step by Step Overview:

1] import all necessary libraries & data.
2] importing the data shuffling it & splitting into 2 sets, a training set & a validation set.
3] we turn the data into numpy arrays so we can do the necessary operations.
4] Initialize random parameters.
5] Define ReLU & softmax function, onehot, & ReLU derivative.
6] Forward propagation
7] Backward propagation

Zaina Abushaban

# Forward propagation :-

$a^{L}$ ← last layer

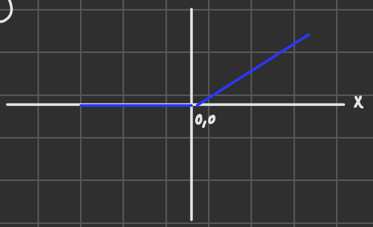
* here we can use ReLU or sigmoid, in the program I used ReLU

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

$$a(L) = \sigma(z^{(L)})^{*}$$

→ using this step we can go from layer to the next layer.

ReLU(x)



→ For the last layer we'll put it through Softmax.

$$\text{ReLU}(x)\begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

## Calculating loss :-

$$\begin{bmatrix} 0.02 \\ 0.1 \\ 0.05 \\ 0.07 \\ 0.89 \\ \vdots \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \text{cost function}$$

Predicted

True [one hot format]

$$\boxed{C_0 = a^{L} - y}$$

This will be the difference, this function represents $C_0$. Which we will need to derive later to find the rate of change of it with respect to weights & biases from the previous layer.

Zaina Abushaban

# Backward propagation

⇒ This step will be repeated for each layer.
such that the cost for the last layer was how we calculated it in the previous page.
For layers before the activation of that layer will act as the cost function
* which means steps will be repeated recursively.

## ⇒ Effect of change in weight to change in cost:

$\frac{\partial C_0}{\partial w^L}$ By the chain rule:-

$$\frac{\partial z}{\partial w} \frac{\partial a^L}{\partial z^L} \frac{\partial C_0}{\partial a^L}$$

$$\boxed{\begin{aligned} C_0 &= a^L - y \\ z^{(L)} &= w^{(L)} a^{(L-1)} + b^{(L)} \\ a^{(L)} &= \sigma(z^{(L)})^{*} \end{aligned}}$$

* from forward prop. & loss calculation

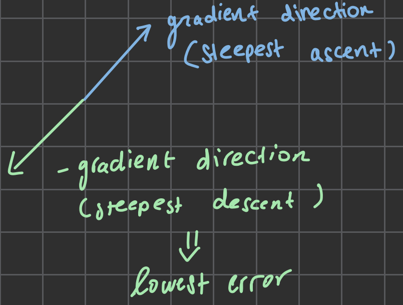
$$\frac{\partial z}{\partial w} = a^{L-1}$$

$$\frac{\partial a}{\partial z} = \sigma'(z^{(L)}) \quad \text{OR} \quad \text{ReLU}'(z^{(L)})$$

→ The same can be done using $\frac{\partial C_0}{\partial b^L}$ and $\frac{\partial C_0}{\partial a^L}$

→ using $\partial a^L$ we can adjust $w^{L-1}$ & $b^{L-1}$

⇒ This way we'll subtract these $\partial$'s from their original values.



← calculating change from gradient values

Zaina Abushaban