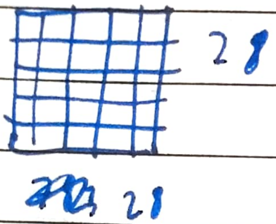
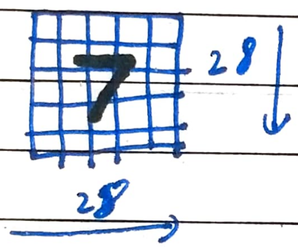
Math



⇒ Each image has 784 pixels

Maths



⇒ Each img. = 28x28 = 784 px.

- px values : 0 – 255
  - 0 ↓ Black
  - 255 ⇒ White

- m no. of images

- Each img is classified : {0, 1, 2, ——, 9}
  ↓
  10 Labels. / classes.

S1: Let's represent it as matrix

- Or, CSV in our case

$$
\underset{\substack{\text{whole}\\ \text{data set}}}{X} : \quad
\begin{array}{c|c}
 & \longleftarrow 784 \text{ values} \longrightarrow \\
\hline
m &
\begin{bmatrix}
— x^{1} — \\
— x^{2} — \\
\vdots \\
— x^{m} —
\end{bmatrix}
\end{array}
$$

$\longleftrightarrow$: Each row is an eg.

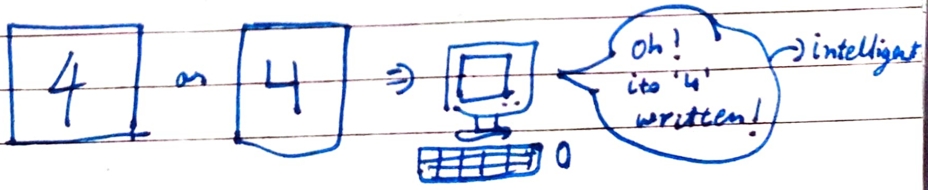
- transpose it

$$
X^{T} = \quad
\begin{array}{c|c}
 & \longleftarrow m \longrightarrow \\
\hline
784 &
\begin{bmatrix}
| & | & & | \\
x^{1} & x^{2} & \cdots & x^{m} \\
| & | & & |
\end{bmatrix}
\end{array}
$$

: Each column $\updownarrow$ is an eg.

* Our goal ?

Take image $\longrightarrow$ Let machine see it $\downarrow$ Predict what is written?



* How ?

⇒ Make a brain

↳ BRAIN

↓

Network of neurons

↓

neural network !

- A quick, simple N.N. : 2+1 layers

1 : input layer : 784 nodes

2 : [1 hidden, 1 output] layer

↓

↳ 10 nodes each

↓

10 nodes / units

↓

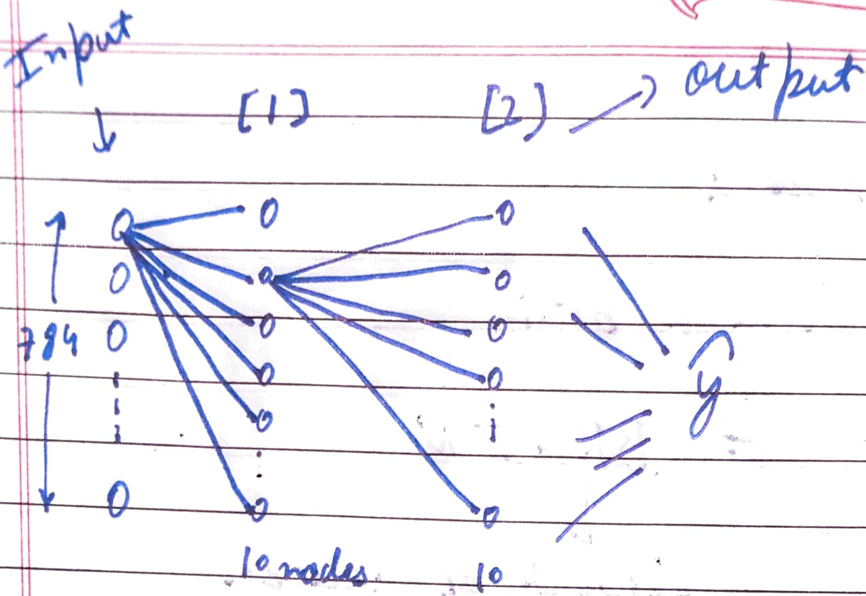
each to one class

↳ 3 options for now, many more available

↳ Leaky ReLu + ELU

↳ only ReLU [ReLu + ReLU]

↳ Sigmoid [Sigmoid + Sigmoid]



Our network

How do we train / make the machine learn?

ⓘ Forward propogation

- Simply, run / pass our img through the network

i.a) take variable ~~A[0]~~ $A^{[0]}$

↳ simply input

$$A^{[0]} = X \quad \text{(the first layer)}$$

i.b) Find $Z^{[1]}$

→ This is un-activated 1^{st} Hidden layer.

$$\underbrace{Z^{[1]} = \underset{10 \times 784}{W^{[1]}} \cdot \underset{784 \times m}{A^{[0]}} + \underset{10 \times 1}{b^{[1]}}}_{10 \times m}$$

- Now, If we directly use $Z^{[1]}$

→ it is not activated, so, we apply activation function

→ ~~So~~ So, lets activate it ?

Lets use, ReLU

★ ReLU.

Rectified, Linear Unit.

↳

$$\text{ReLU}(x) \begin{cases} x \geq 0 = x \\ x \leq 0 = 0 \end{cases}$$

Simple, but adds complexity we need.

i.e. Let's apply activation function

$g(x)$

our function right now? ⇒ ReLU.

Hence :

$$A^{[1]} = g[Z^{[1]}] = \text{ReLU}[Z^{[1]}].$$

↓
Layer 1

Now, similarly From layer 1 → layer 2

i.d) Find $Z^{[2]}$

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

$$10 \times 10 \quad 10 \times m \quad 10 \times 1 \Rightarrow 10 \times m$$

i.e). Activate layer 2

$g(x)$; lets use Softmax

Hence ; $A^{[2]} = \text{Softmax}(Z^{[2]})$

⊙ Softmax ⇒ will give a probability

(6) So now, we made a prediction

But ? it might be wrong.

- Hum kya krte hein?

→ Apni galti se seekho....

ii). Back - Propogation.

- See how much error?
- See weights / biases 's contribution to the error
- Adjust them to minimize the error.

$dz^2 : A^{[2]} - Y$ // How far from Label

$d[W]^2 : \frac{1}{m} dz^{[2]} . A^{[1]T}$

$d[b]^2 : \frac{1}{m} \sum dz^{[2]}$

Similarly, for Layer 1

$$dz^{[1]} : W^{[2]T} \cdot dz^{[2]} .* g'[z]$$

derivative of activation function

$$dW^{[1]} : \frac{1}{m} dz^{[1]} \cdot X^T$$

$$db^{[1]} : \frac{1}{m} \Sigma dz^{[1]}$$

iii) Update the Parameters

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

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

Same; $W^{[2]}$, $b^{[2]}$.

α : Learning rate.

→ Hyper parameter, You set it.