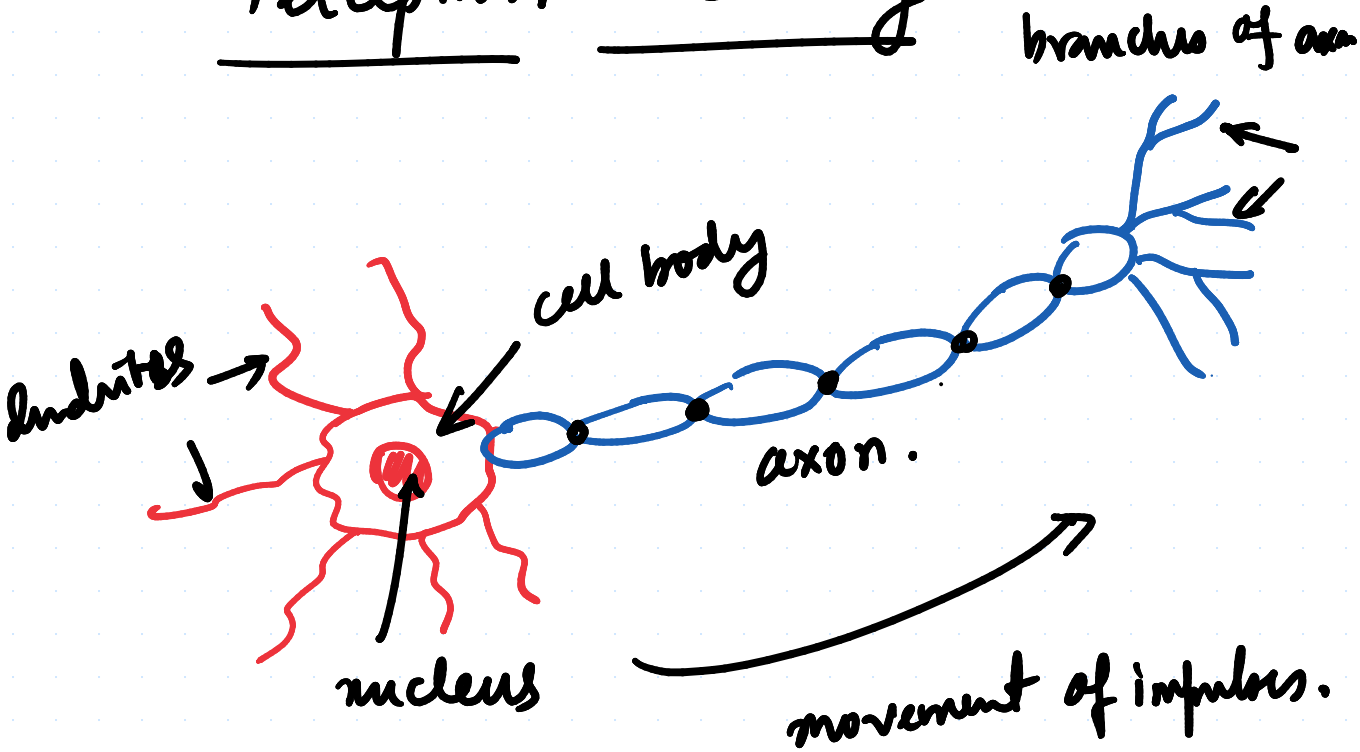
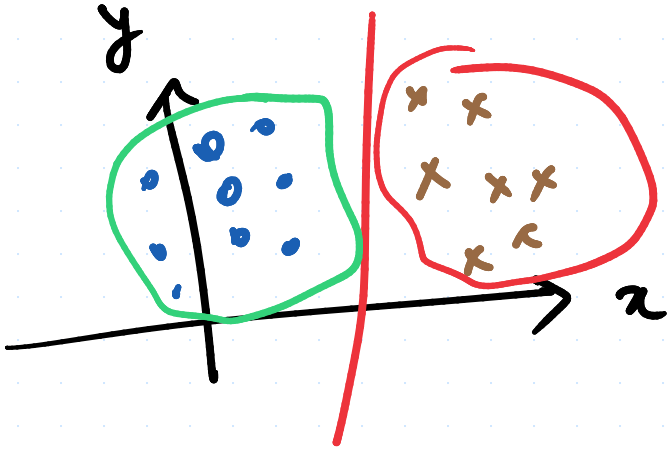
Lecture-3

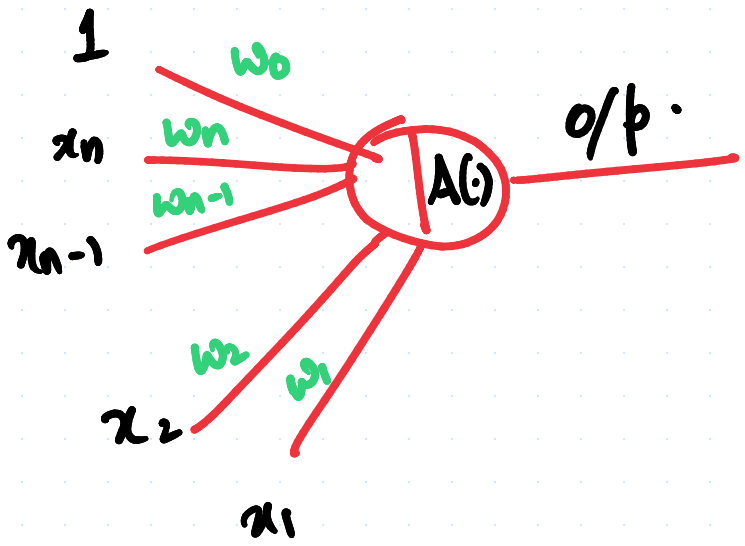
01/02/26.

Perceptron - learning



* Neuron only fires when the sum of the weighted signals is greater than a threshold.





$1 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 + \cdots + x_{n-1} w_{n-1} + x_n w_n$.

$$A\left(\sum_{i=1}^{D} w_i x_i + w_0\right)$$

Perceptron is more similar to single layer feed-forward neural networks & only one layer of weights is used.

Training algorithm.

correct classification

if $x \in y = 1$, then $w^T x > 0$

if $x \in y = -1$, then $w^T x < 0$.

Two equations can be jointly written as.

$$y w^T x > 0.$$

Functional margin $\hat{\gamma}_n$ of an example $(x^{(n)}, y^{(n)})$ w.r.t. hyperplane is:-

$$\hat{\gamma}_n = y^{(n)} (w^T x^{(n)})$$

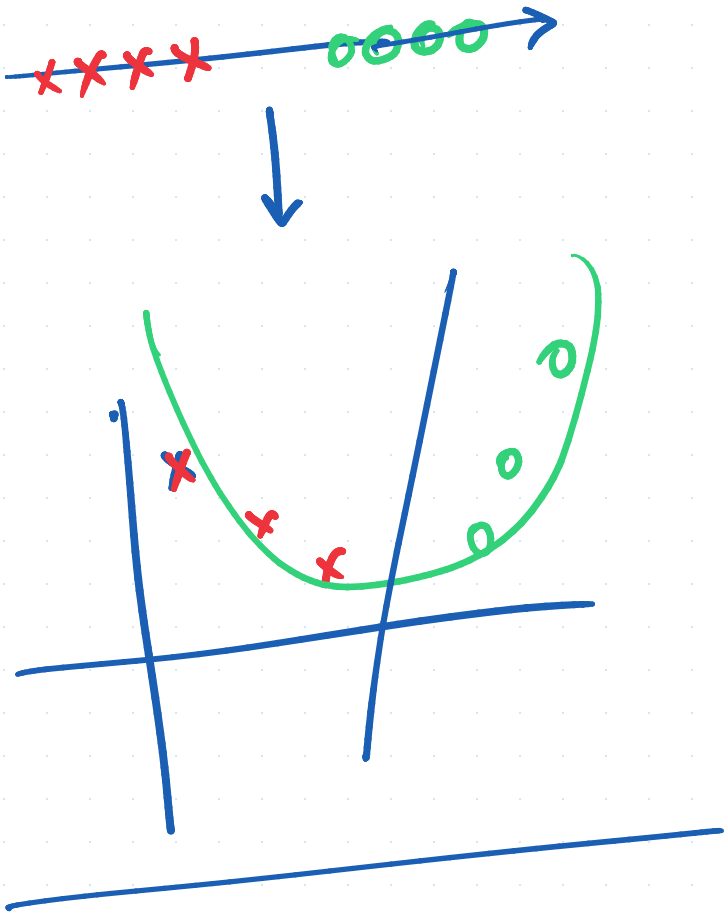
+ve $\hat{\gamma}_n \rightarrow$ example is correctly classified

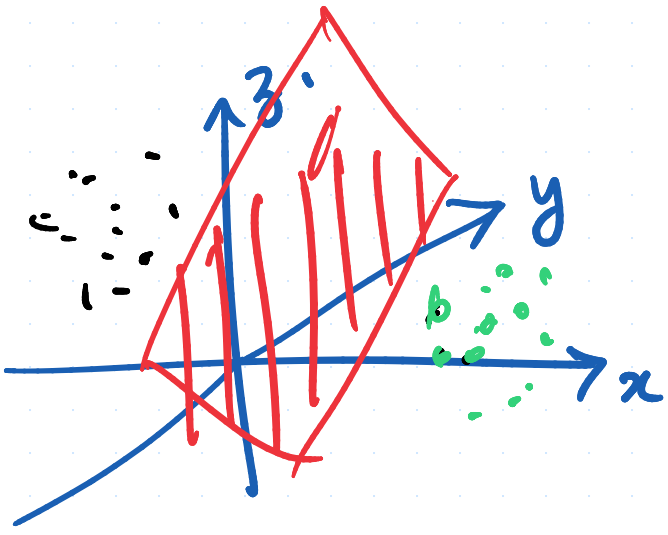
-ve $\hat{\gamma}_n \rightarrow$ example is incorrectly classified

Geometric Margin:

γ_n of an example → signed ⊥ distance of the point to the hyperplane.

$$\gamma_n = \frac{w^T x^{(n)}}{\|w\|}.$$





$\gamma = \min_{D} |\gamma_n|$. → minimum of the geometric margin.

Perceptron algorithm checks if all the training examples are correctly classified w.r.t. a hyperplane.

→ in case of misclassification, the hyperplane is updated.

After k-updates, a misclassified example $(x^{(n)}, y^{(n)})$ is encountered, then $w^{(k)}$ is updated as:-

$$w^{(k+1)} = w^{(k)} + y^{(n)} x^{(k)}.$$

if $(x^{(n)}, y^{(n)})$ is misclassified after k-update to the weights. $\gamma_n = y^{(n)} (w^{(k)})^T x^{(n)} < 0$.

$$w^{(k+1)} = w^{(k)} + y^{(n)} x^{(n)}.$$

Then the new margin is then,

$$\gamma_n^{(new)} = y^{(n)} \left(w^{(k+1)}\right)^T x^{(n)}$$

$$= y^{(n)} \left(w^{(k)} + y^{(n)} x^{(n)}\right)^T x^{(n)}$$

$$= y^{(n)} \left(w^{(k)T} x^{(n)}\right) + \left(y^{(n)}\right)^2 \cdot \|x^{(n)}\|^2$$

$$\geq y^{(n)} (w^{(k)})^T x^{(n)}$$

$$\geq \gamma_n^{(old)}$$

the new hyperplane $w^{(k+1)}$ has a larger

margin than the older one ⇒ better classification of $(x^{(n)}, y^{(n)})$

Algorithm :-

$w = 0$: initialize.

while (not converged) then

$k = 0$

for $n = 1, 2, \ldots, N$ do.

if $y^{(n)}((x^{(n)})^T x) \leq 0$ then

$w \leftarrow w + y^{(n)} x^{(n)}$

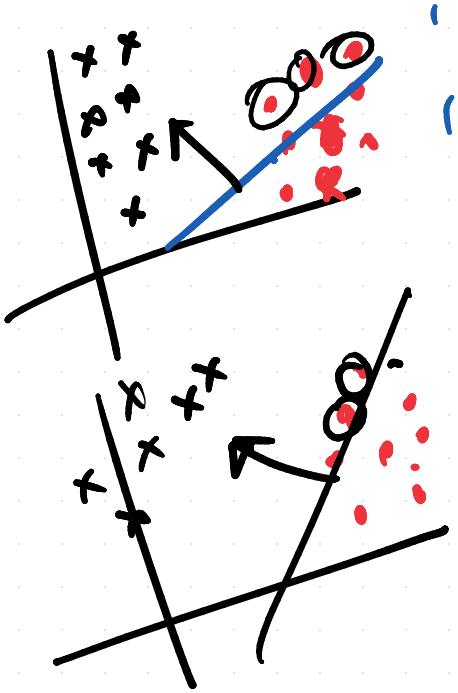
$k \leftarrow k + 1$

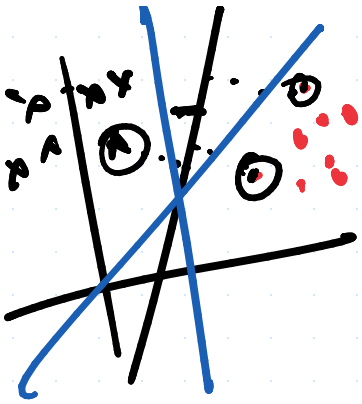
end if

end for

if $k = 0$ then

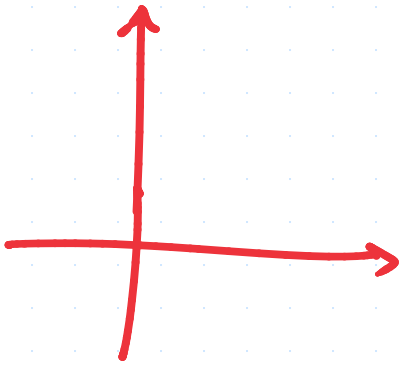
break





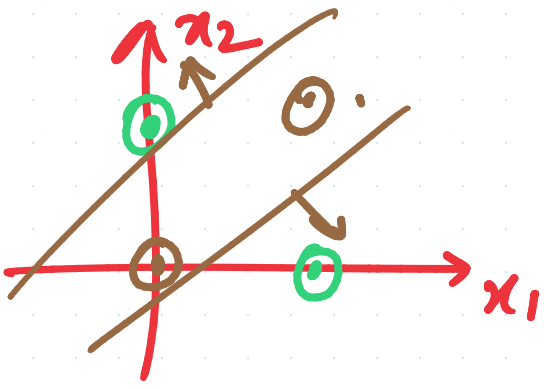
endif.

end loop.

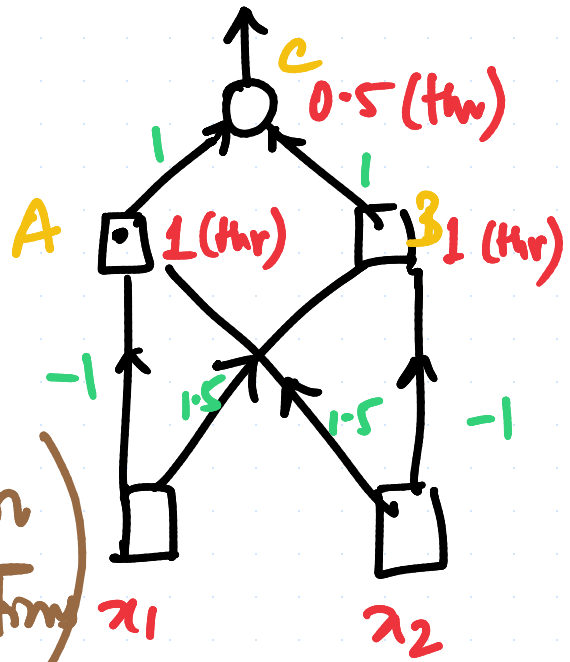


Classification Network.

x_1	x_2	$x_1 \oplus x_2 = x_1 \bar{x}_2 + \bar{x}_1 x_2$
0	0	0
0	1	1
1	0	1
1	1	0



x_1	x_2	A's o/p.
0	0	0
0	1	1
1	0	0
1	1	0



5-neuron
6-connections

x_1	x_2	B's o/p
0	0	0
0	1	0
1	0	1
1	1	0

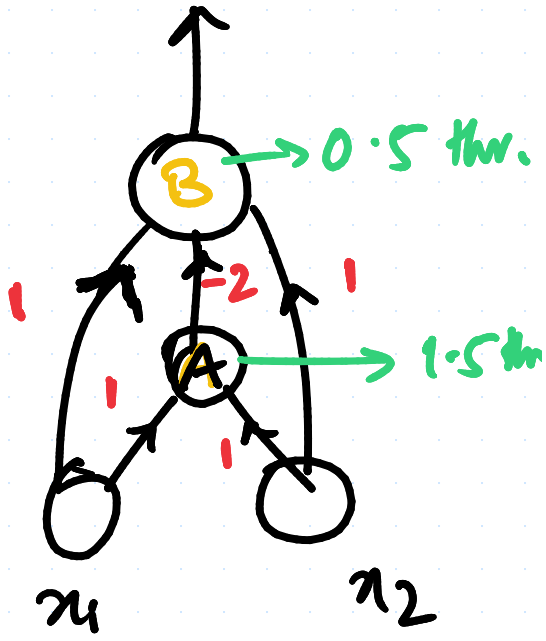
$0 \times (-1) + 1.5 \cdot (0) > 1!$

Multi-layer perceptron classifies XOR.

A's	B's	C's o/p.
0	0	0
1	0	1
0	1	1
0	0	0

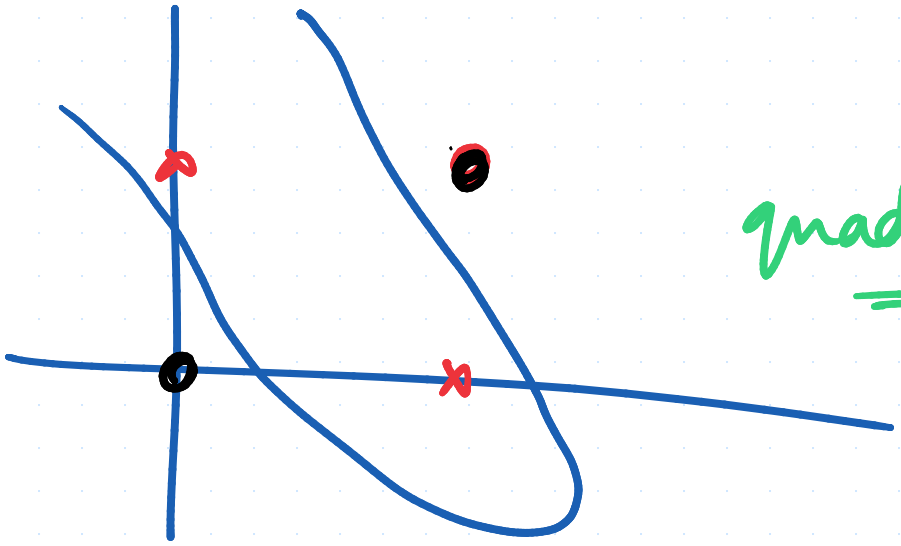
x_1	x_2	A's	B's	o/p
0	0	0	0	
0	1	0	1	
1	0	0	1	
1	1	1	0	

XOR.

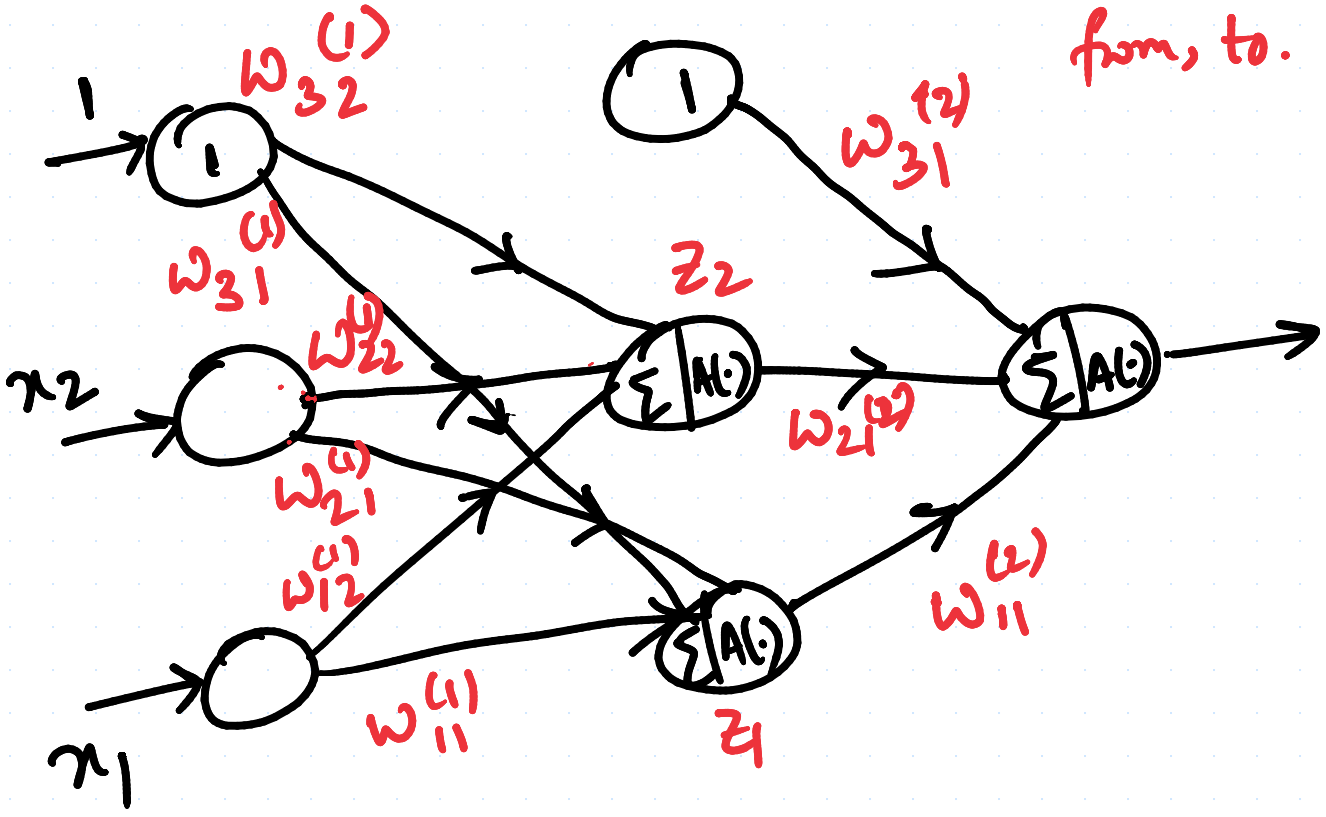


4-neuron

5-connection.



quadratic activation



$$z_1 = w_{11}^{(1)} \cdot x_1 + w_{21}^{(1)} \cdot x_2 + w_{31}^{(1)} \cdot 1.$$

$$z_2 = w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{32}^{(1)} 1.$$

$$\begin{cases} w_{11}^{(1)} = 1, \quad w_{21}^{(1)} = 1, \quad w_{31}^{(1)} = 0.5 \\ w_{12}^{(1)} = 1, \quad w_{22}^{(1)} = 1, \quad w_{32}^{(1)} = -1.5 \end{cases}$$

x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

o/p.

$z_1 = ?$

$z_2 = ?$

Activation of $z_1 = ?$

Activation of $z_2 = ?$

Guess; Numerical equations,

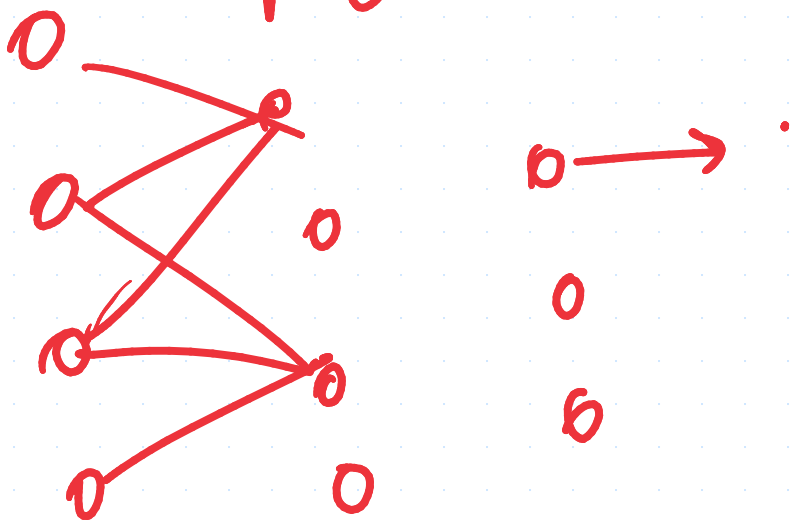
linear equations → values.

NN. → 2 Layer NN.

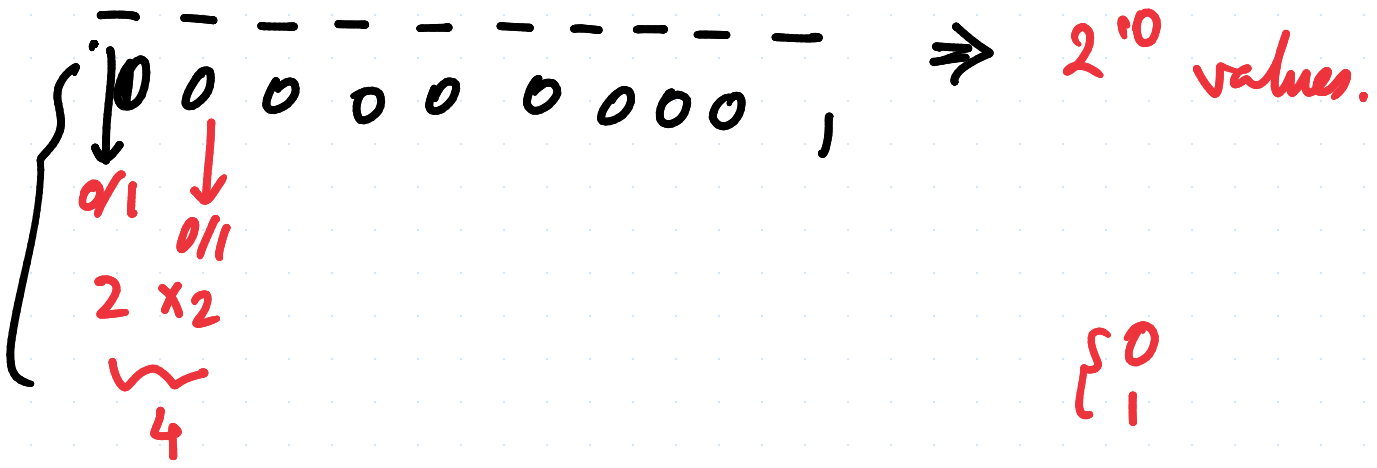
initialise, train;

converge.

F.C-NN.



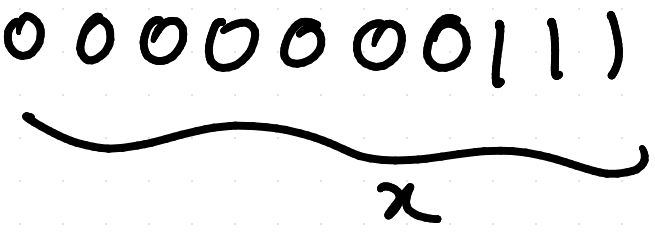
10-digit binary numbers.

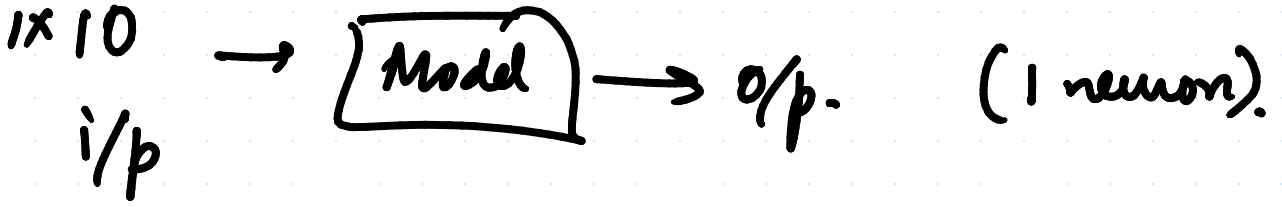


#1 ≥ #0's.?

0 0 0 1 0 1 1 0 1 → 1
x y

1 0 0 0 1 1 1 1 1 1 → 1
x y





Pytorch - pipeline:

- Data → processing (i/p, o/p ; loading etc.
- Model → Pytorch.
- i/p → Model → o/p.
- Visualization / error analysis

DL project

train() → model.train()

validation() → model.eval(). → Test how

test () → model.eval(). → good the model is.

master - epoch - controller.

( )

→ train

→ valid

→ test

logging.

criterion → 50 epoch → save.