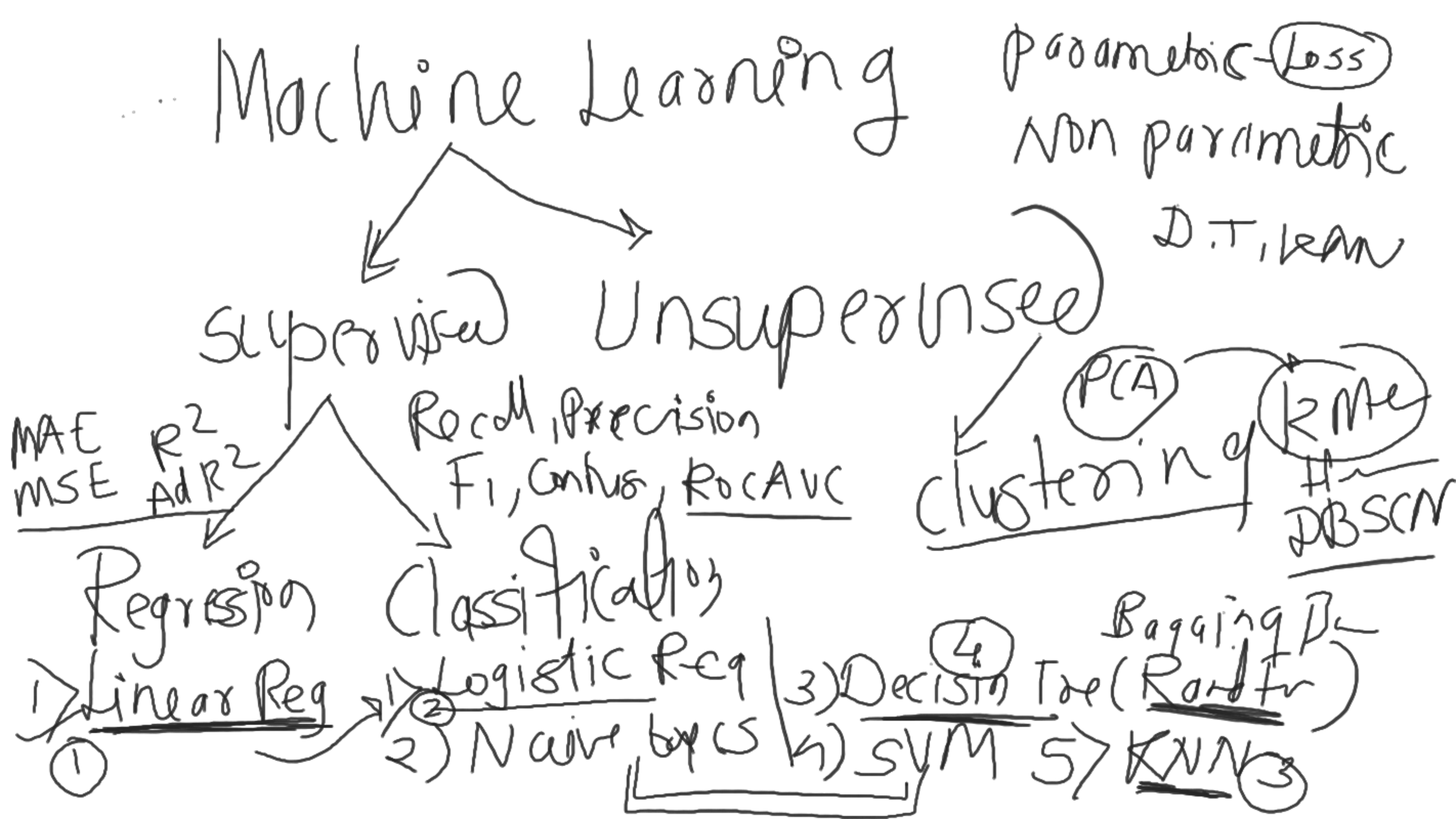
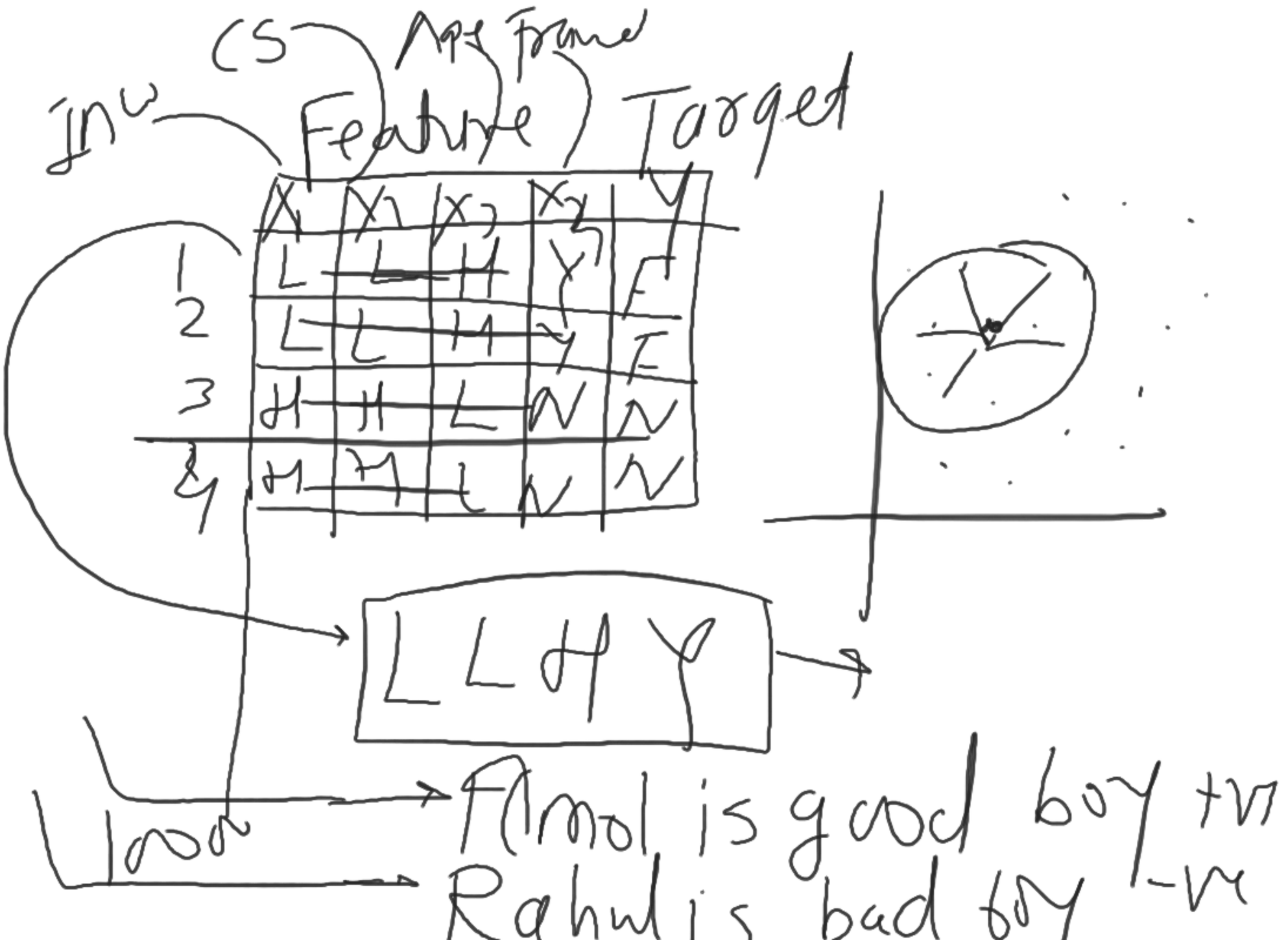
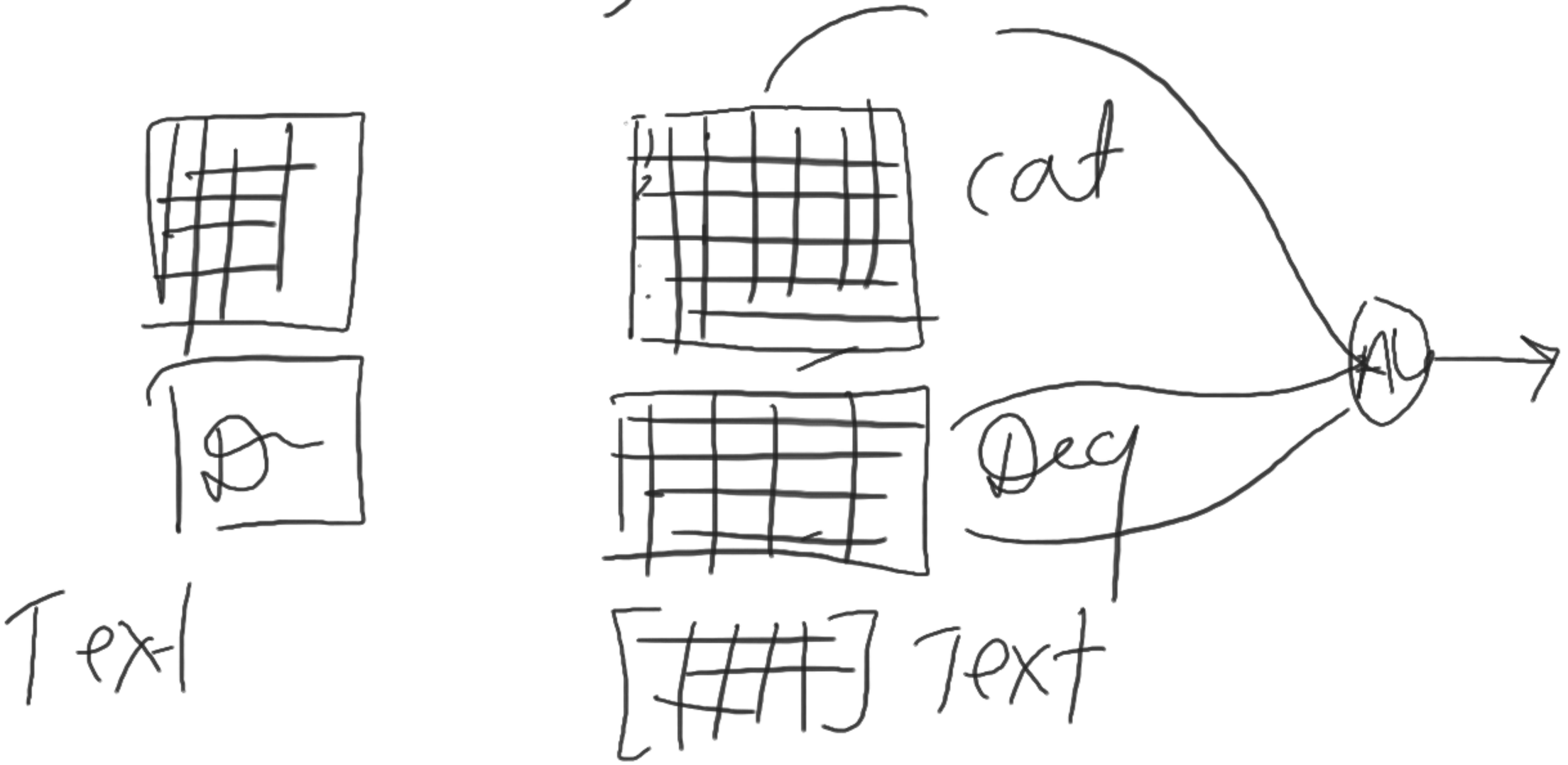

Machine Learning
parametric
loss
Non parametric
D.T, KNN
Supervised
Unsupervised
MAE
MSE
R^2
AdR^2
Recall, Precision
F1, Confus, RocAUC
PCA
KMe
H~
DBSCN
Regression
Classification
Clustering
1) Linear Reg
1)
2)
2) Logistic Reg
Naive bayes
3) Decision Tree (Rand Fr
4)
Bagging B~
4) SVM
5) KNN
3


CS
Age Found
Inc
Feature
Target
1
2
3
4
L L H Y
Amol is good boy +ve
Rahul is bad boy -ve

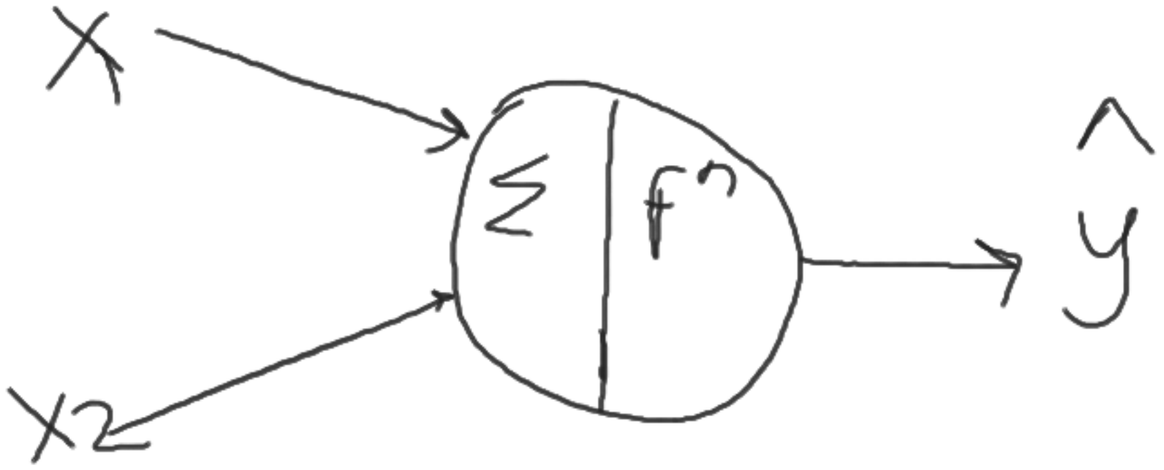
Deep Learning



Biological & Artificial Neurons

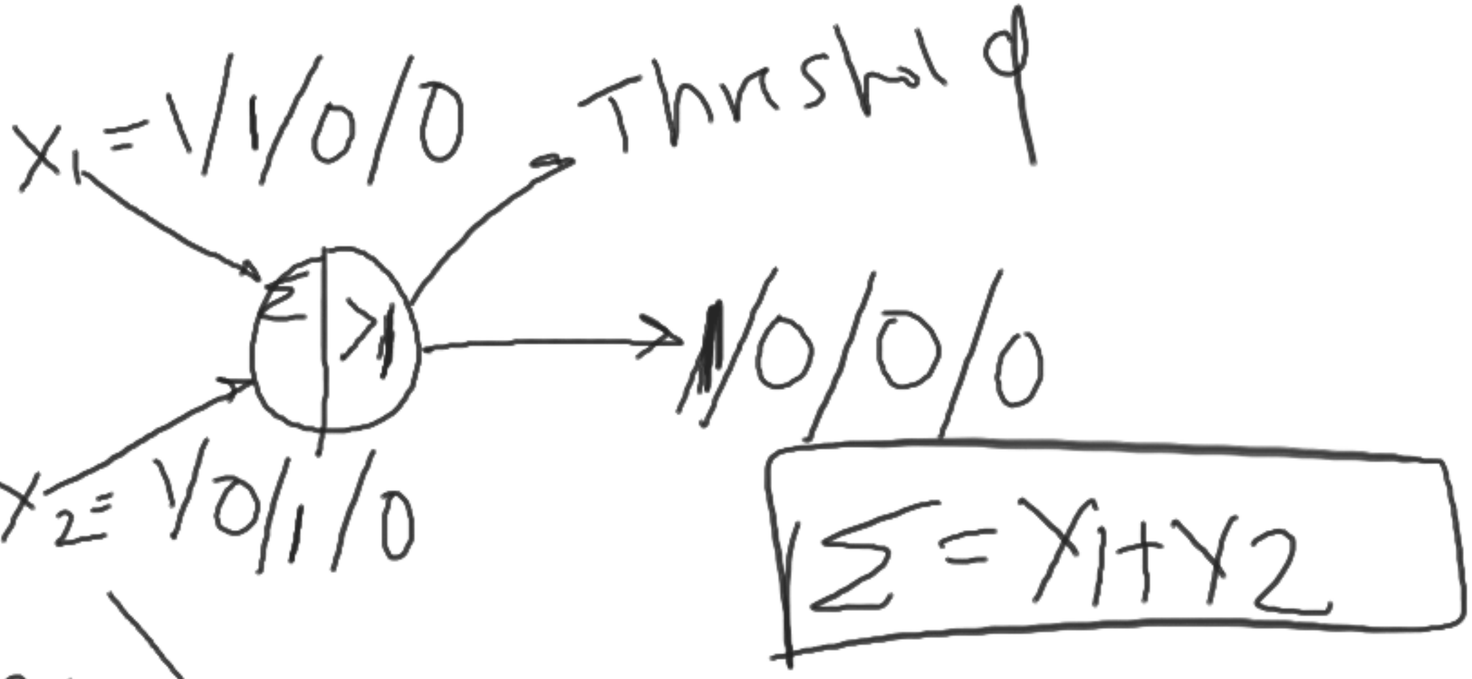
MCPN ⇒ McCulloh & Pits Neuron

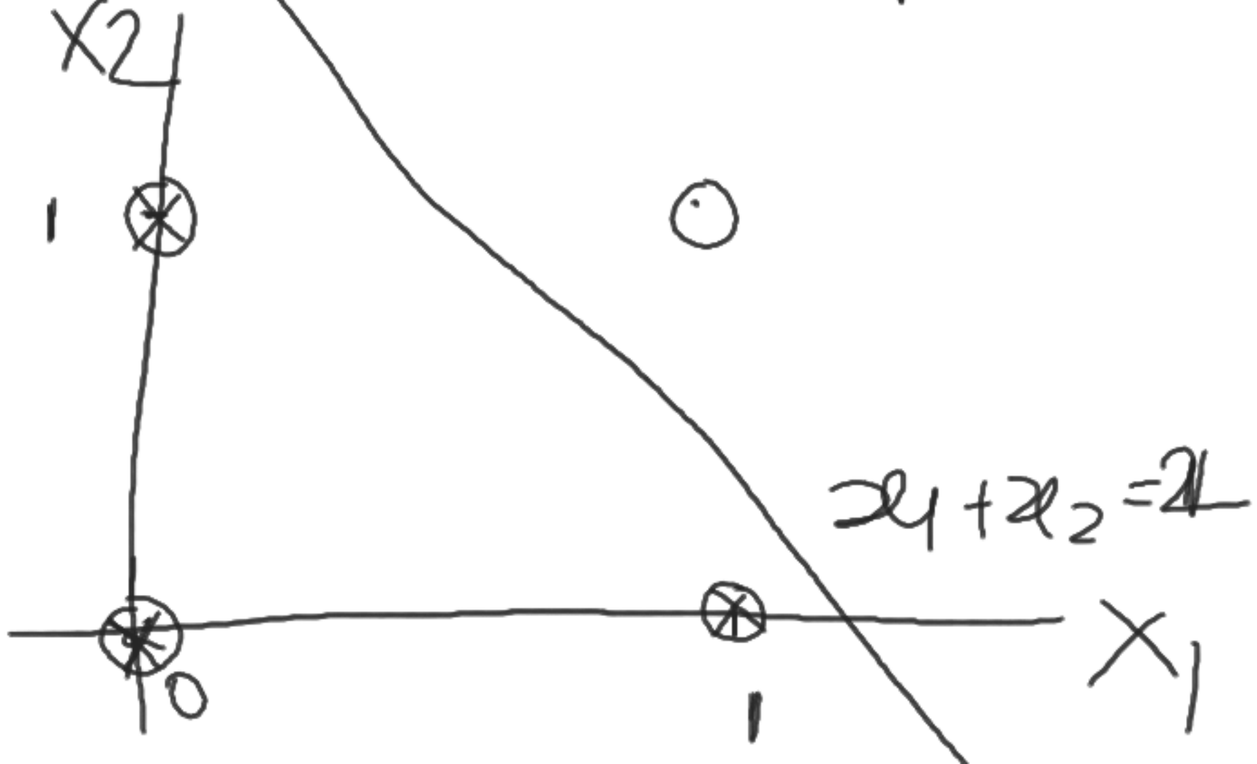
Mimicked the biological neuron & produced mathematical model



AND

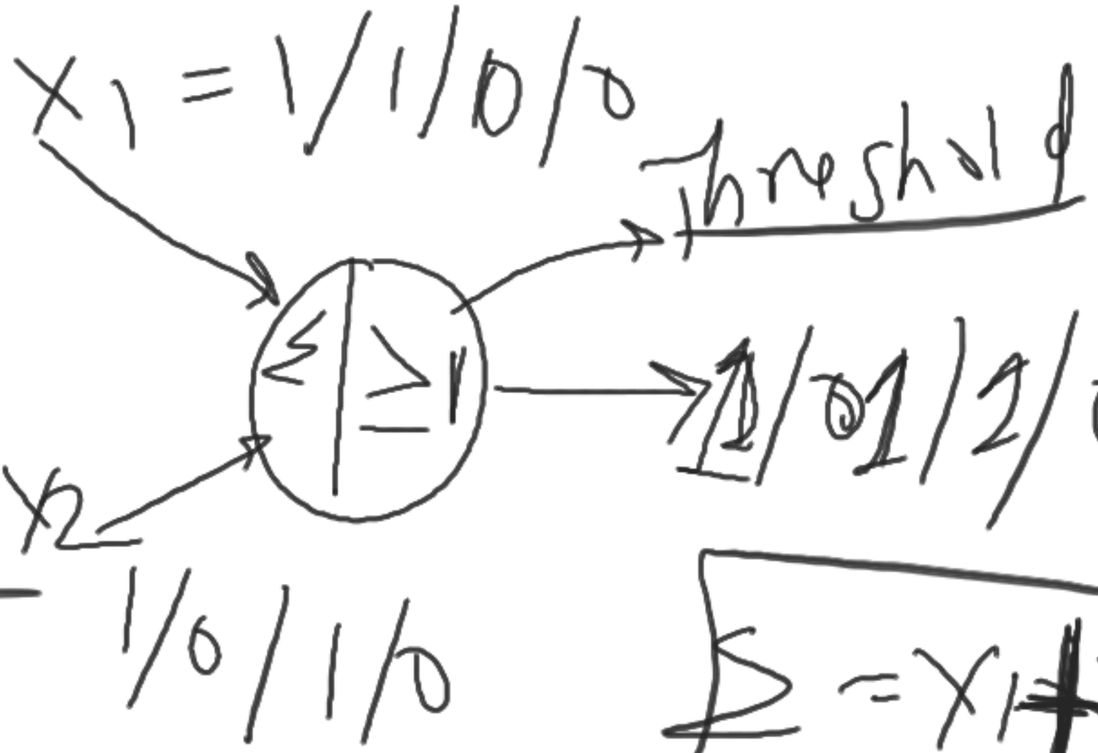
X_1	X_2	y	Sum	
1	1	1	2	1
1	0	0	1	0
0	1	0	1	0
0	0	0	0	0



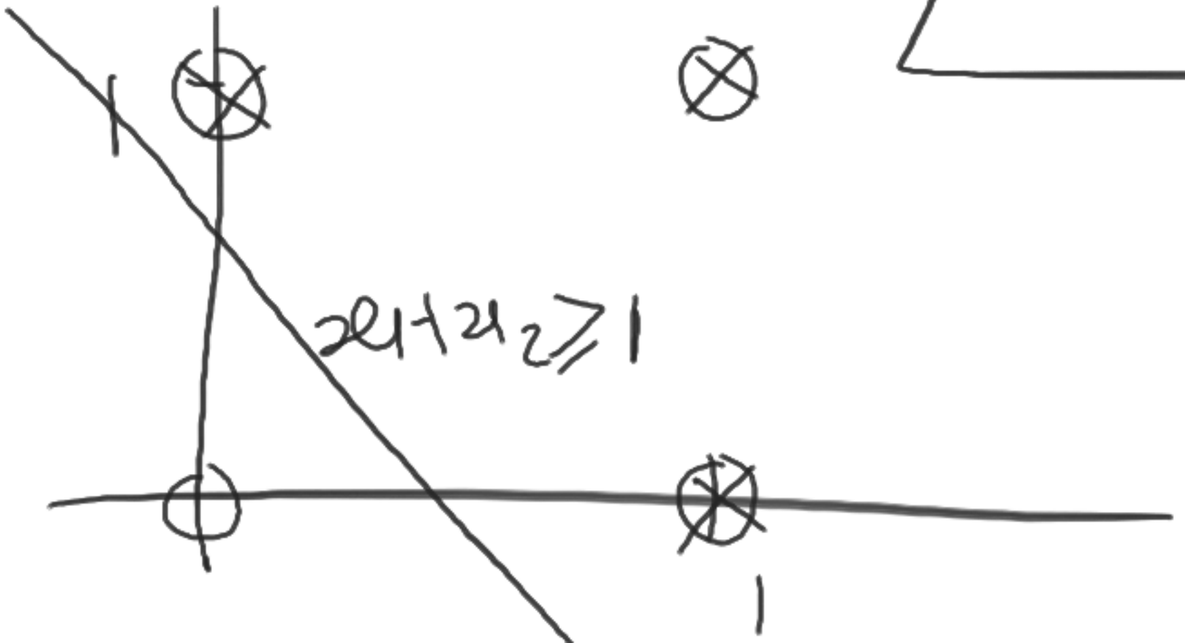


OR

X_1	X_2	y	sum
1	1	1	2
1	0	1	1
0	1	1	1
0	0	0	0



$y = x_1 + x_2$



By changing the threshold of using this methodology (MCPN) we can able to mimics these gates

But problem is that we have to manually change the threshold which need human intervention.

However our objective was without human intervention (can machine learn threshold)

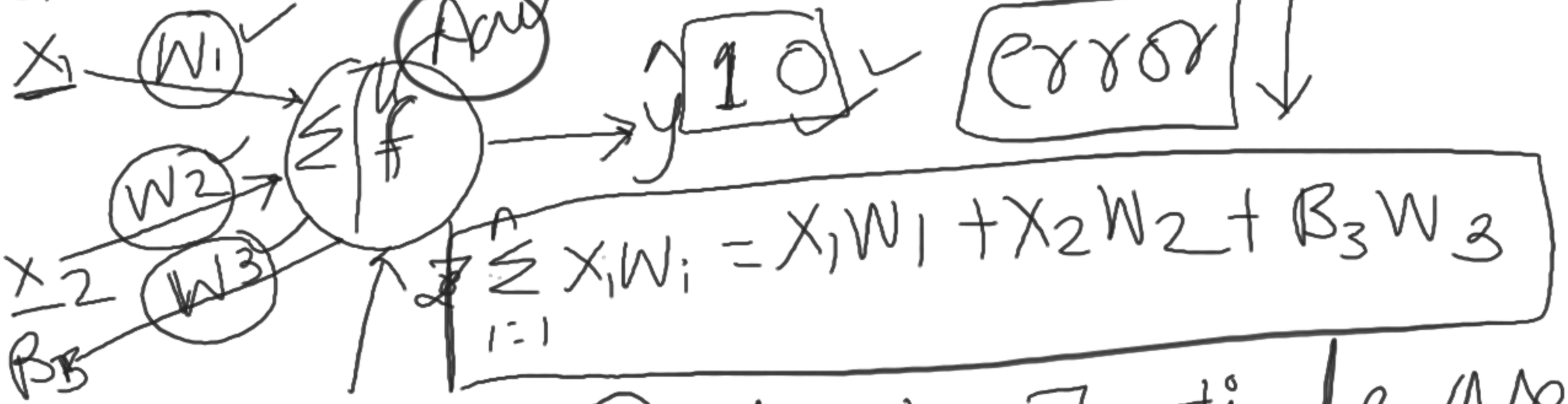
1) ~~Non boolean~~ what if input is non boolean (0/1) ?

2) Do we always need to hand code threshold ?

Rosenblatt → American ~~psy~~ psychologist

proposed the classical perceptron model

Similar to MCPN, but he unneeded weights



① Randomly Initiate Weight

1) Weights initialized randomly & then get adjusted as per error to minimize it

2) It is same as done in linear regression drawing random line

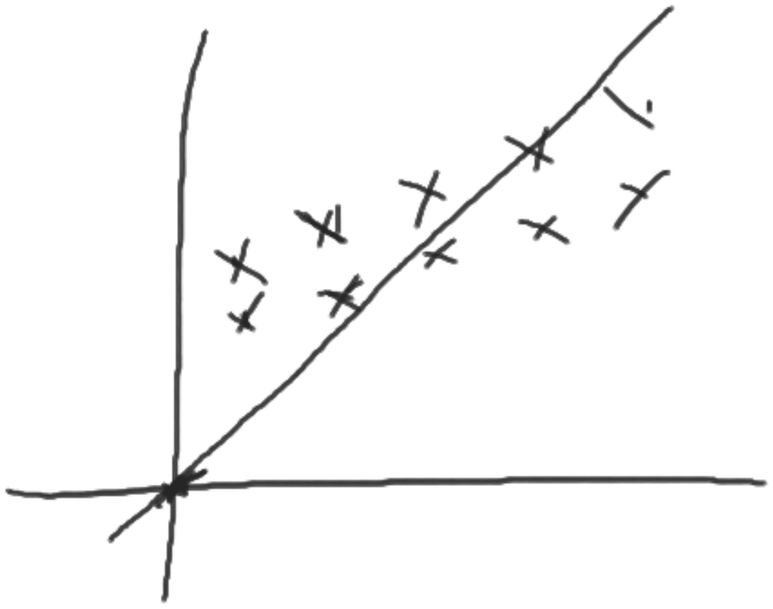
$E_1 = 100$

$E_2 = 20$

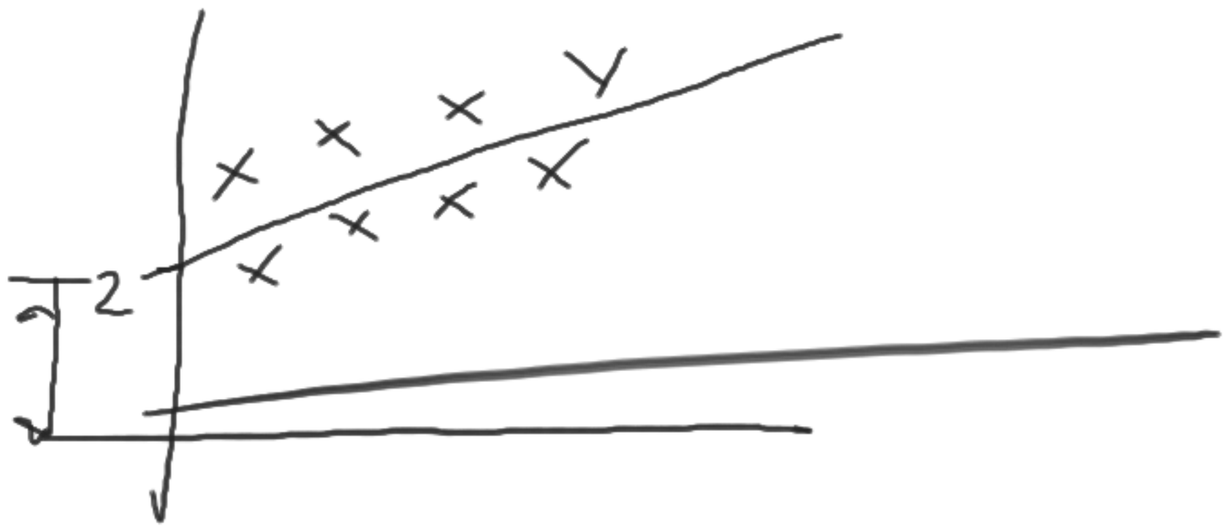
$E_3 = 5$

$E_4 = 10$

3) Bias will provide extra DoF to choose best weight



$y = mx + 0$



$y = mx + c$

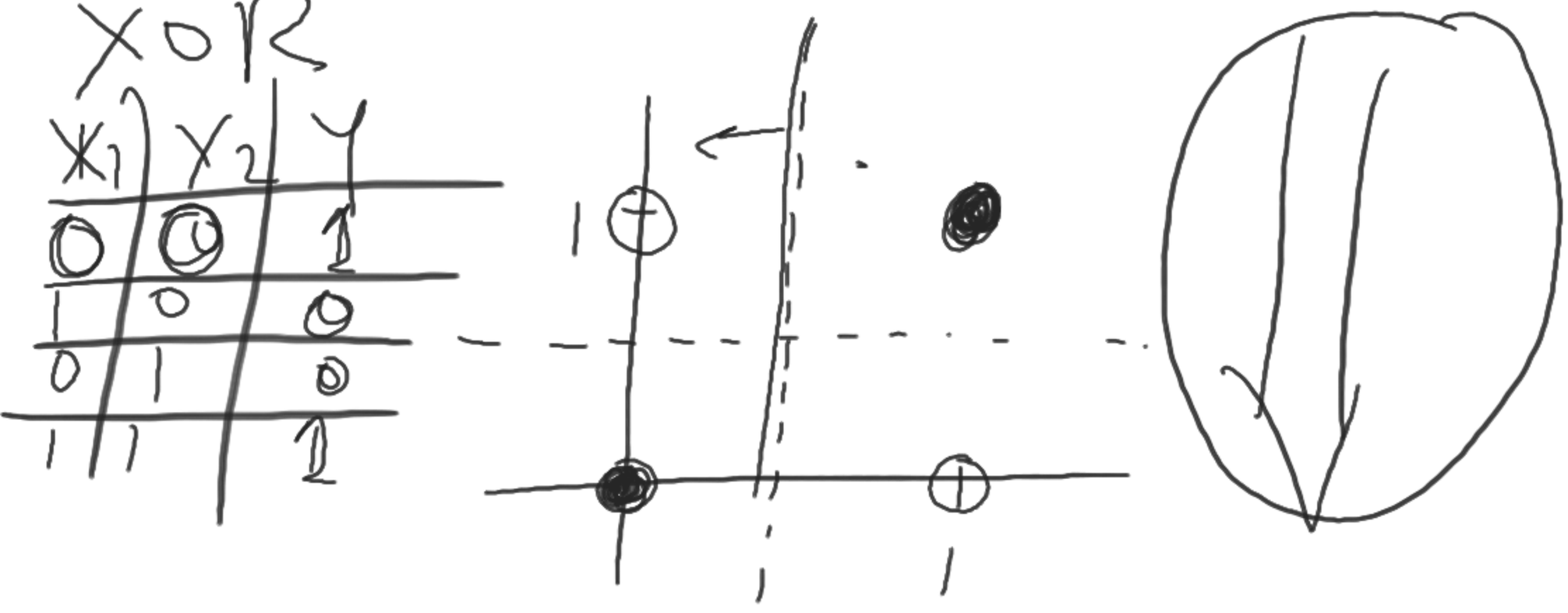
$$error = \sum (\hat{y} - y)^2$$

$$= \sum (mx + c - y)^2$$

However still single perceptron cannot learn all the gates e.g. XOR

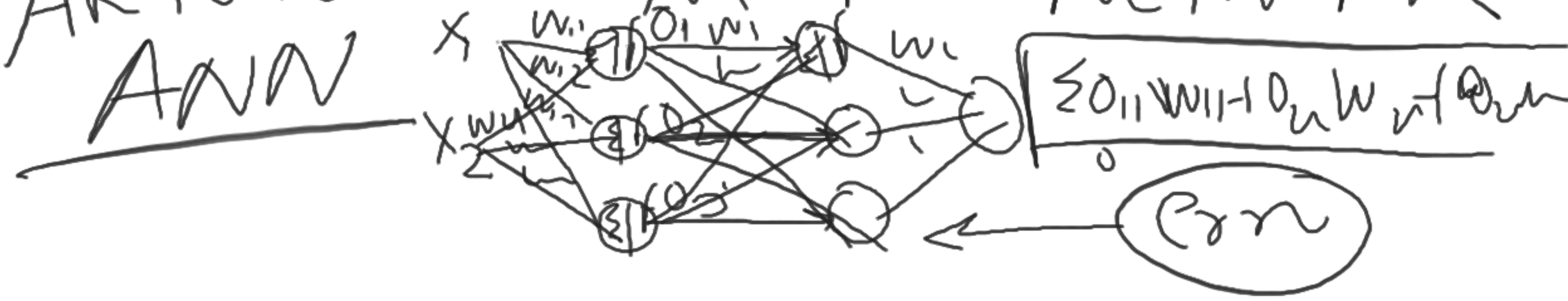
XOR

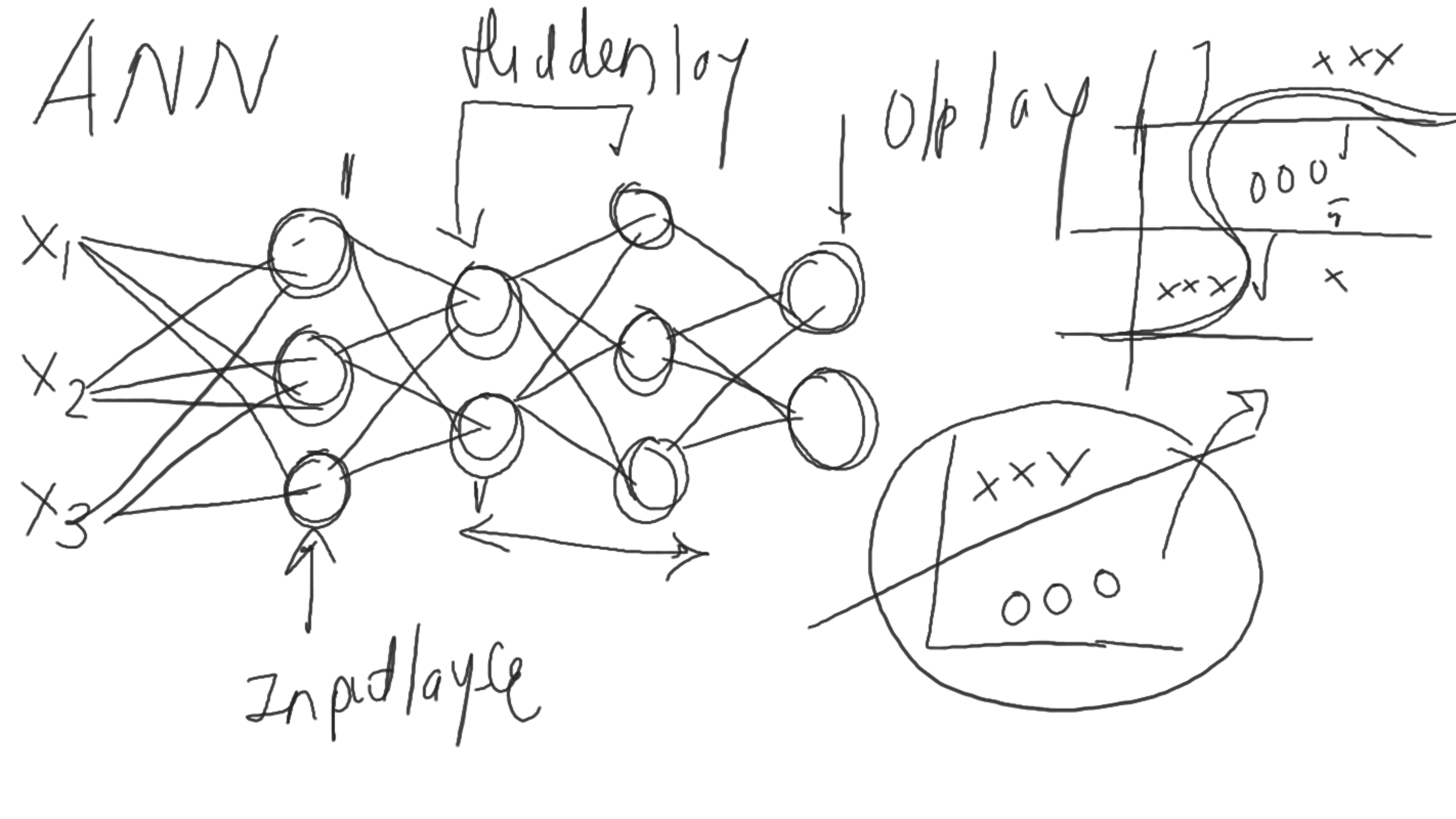
X_1	X_2	Y
0	0	0
1	0	1
0	1	1
1	1	0

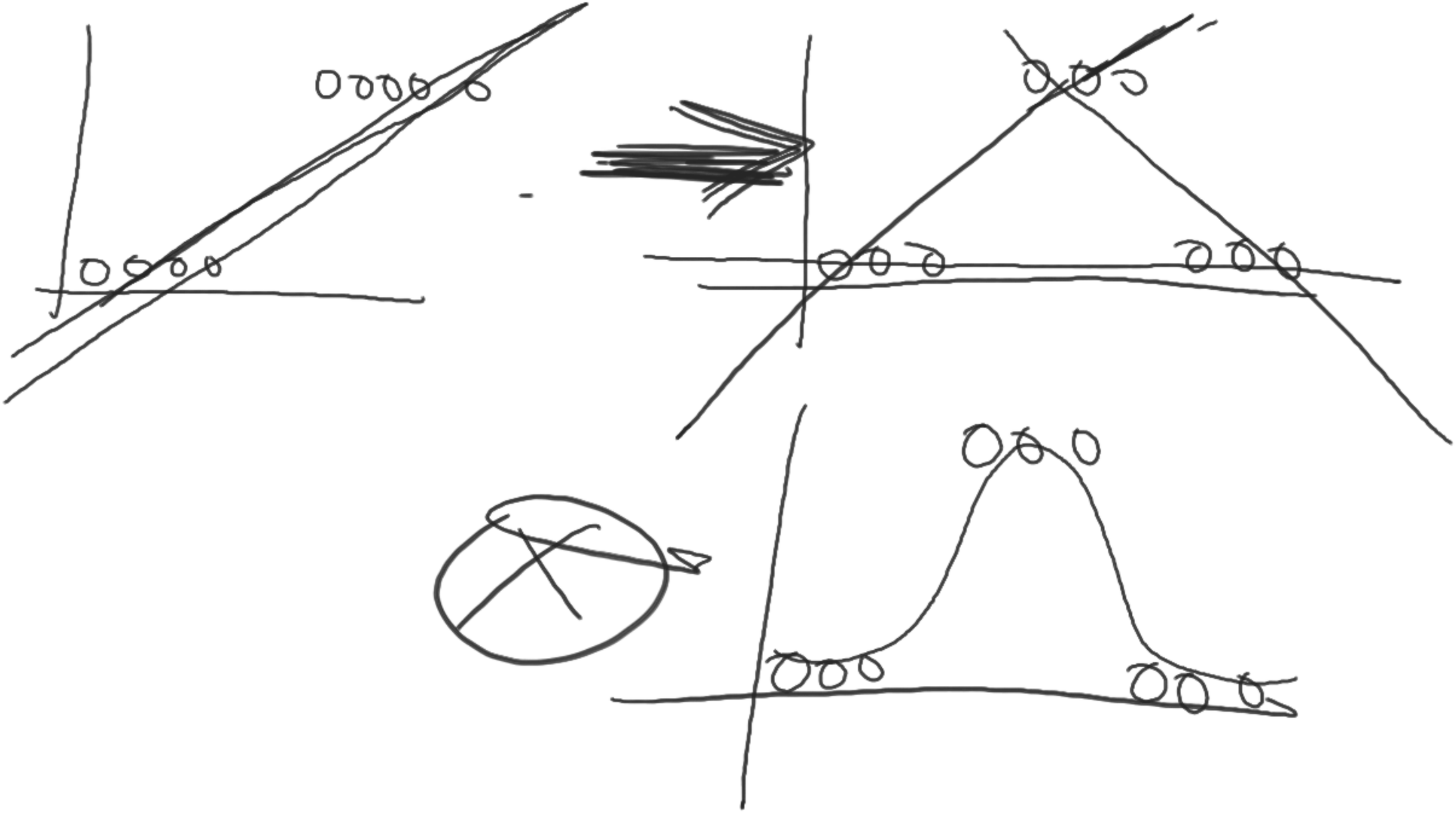


So ~~Instead~~ Instead of making 1 neuron(perceptron) learn all the things, we should make connections of neurons & that connections should learn all the things
This connections of perceptrons is called as ARTIFICIAL NEURAL NETWORK

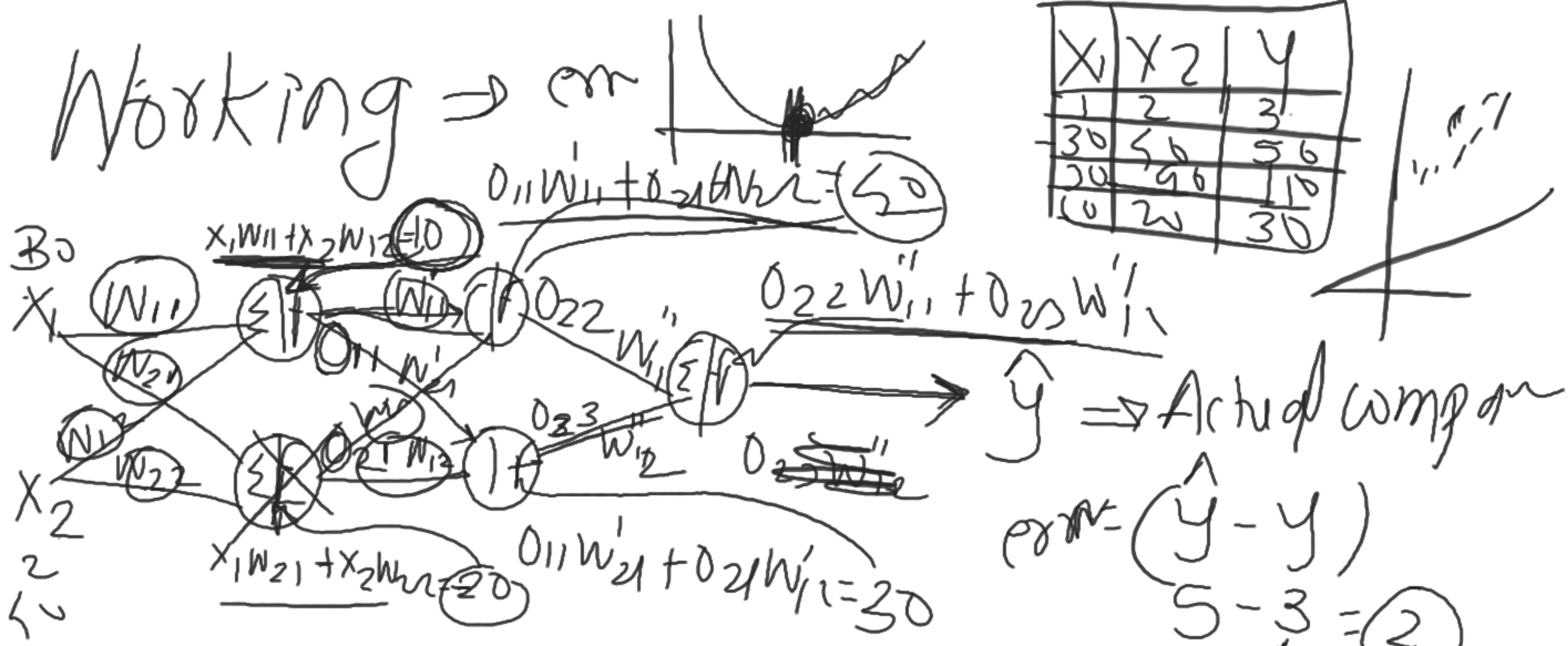
ANN




ANN
Hiddenlay
Oplay
x x x
o o o
x x x
x
x x y
o o o
x_1
x_2
x_3
Inputlayer



Working ⇒ on



X1	X2	Y
1	2	3
30	40	56
20	60	10
10	20	30

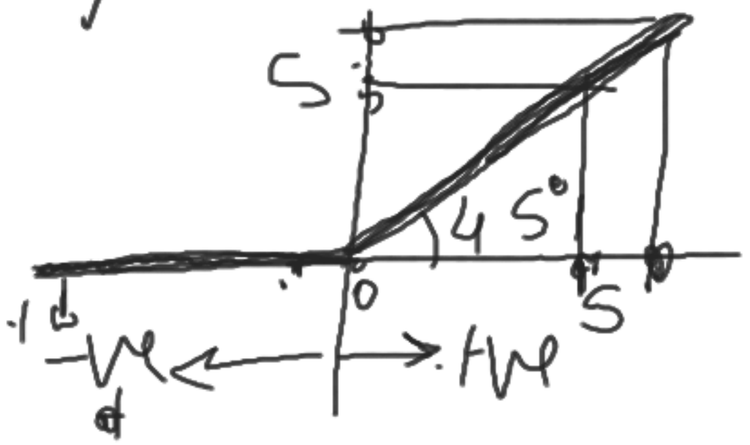
Forward Propogation

Backward

Activation function

- NON linearity
- It decides what should be fired to next neuron
- Non negative!

1) ReLu → Rectified Linear Unit



$$ReLu(x) = max(0, x)$$

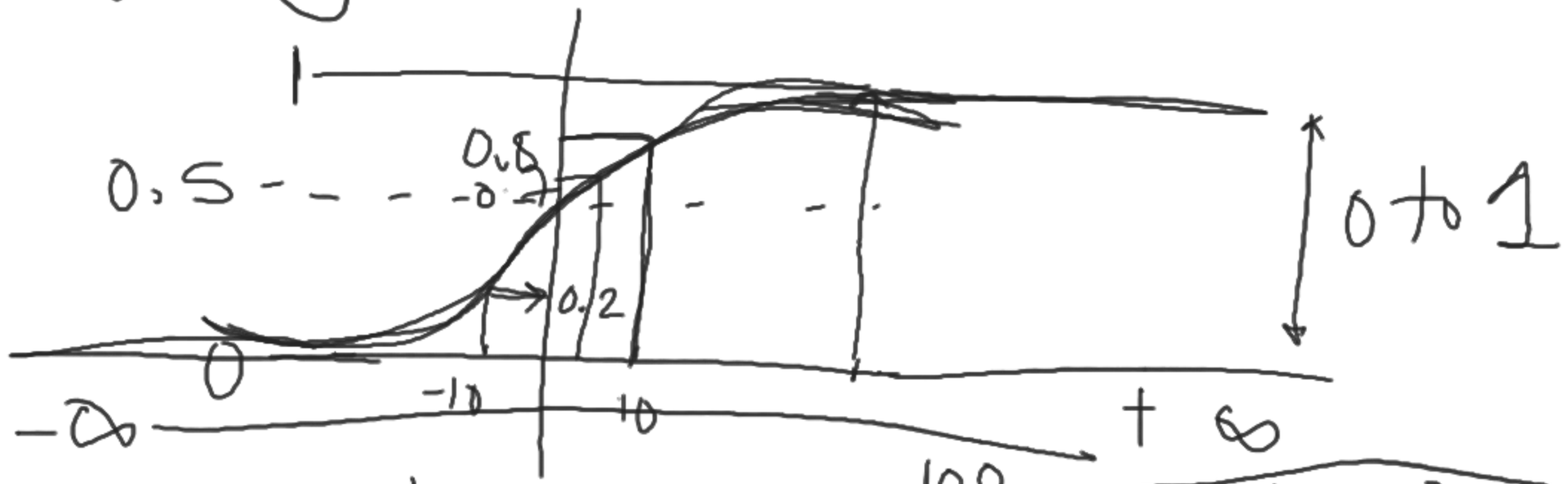
$$Relu(10) = max(0, 10) \rightarrow 10$$

$$Relu(5) = max(0, 5) \rightarrow 5$$

$$Relu(-10) = max(0, -10) \rightarrow 0$$

$$Relu(-100) = max(0, -100) \rightarrow 0$$

2) sigmoid function



3) tanh $\Rightarrow$ $\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$