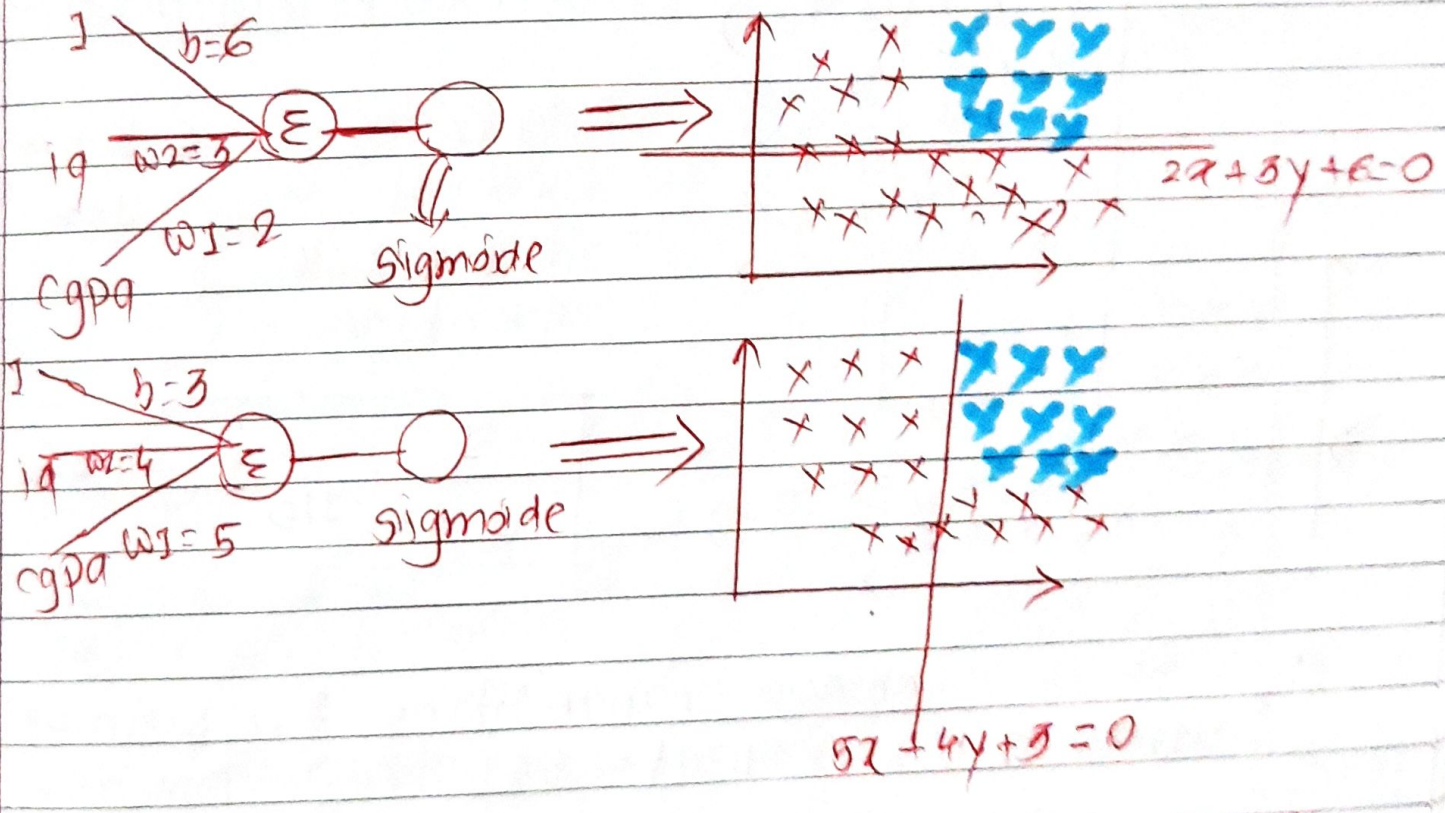
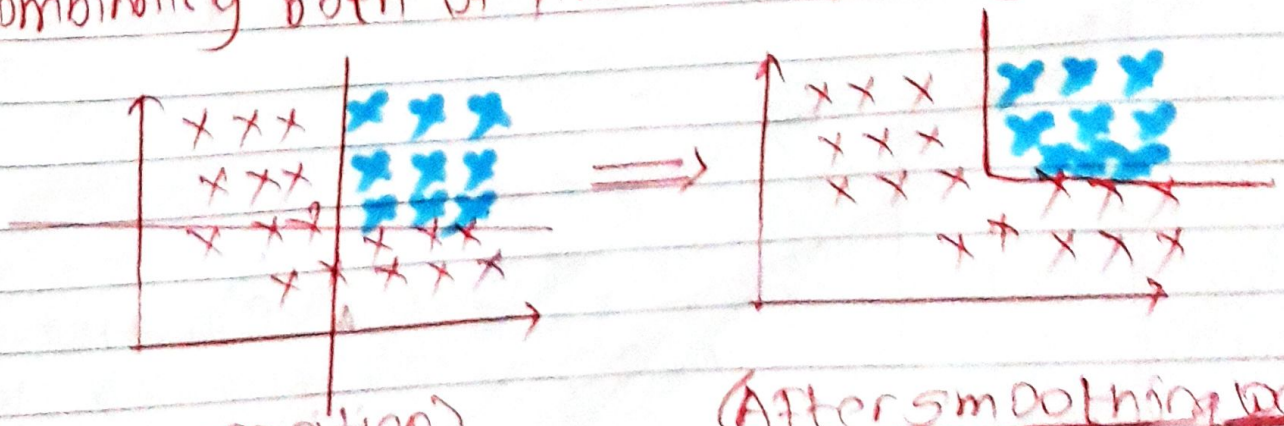
होती है

Let's form a idea of multilayer perceptron

- Apply two perceptron on some dataset.



- By combining both of perceptron output we get this,



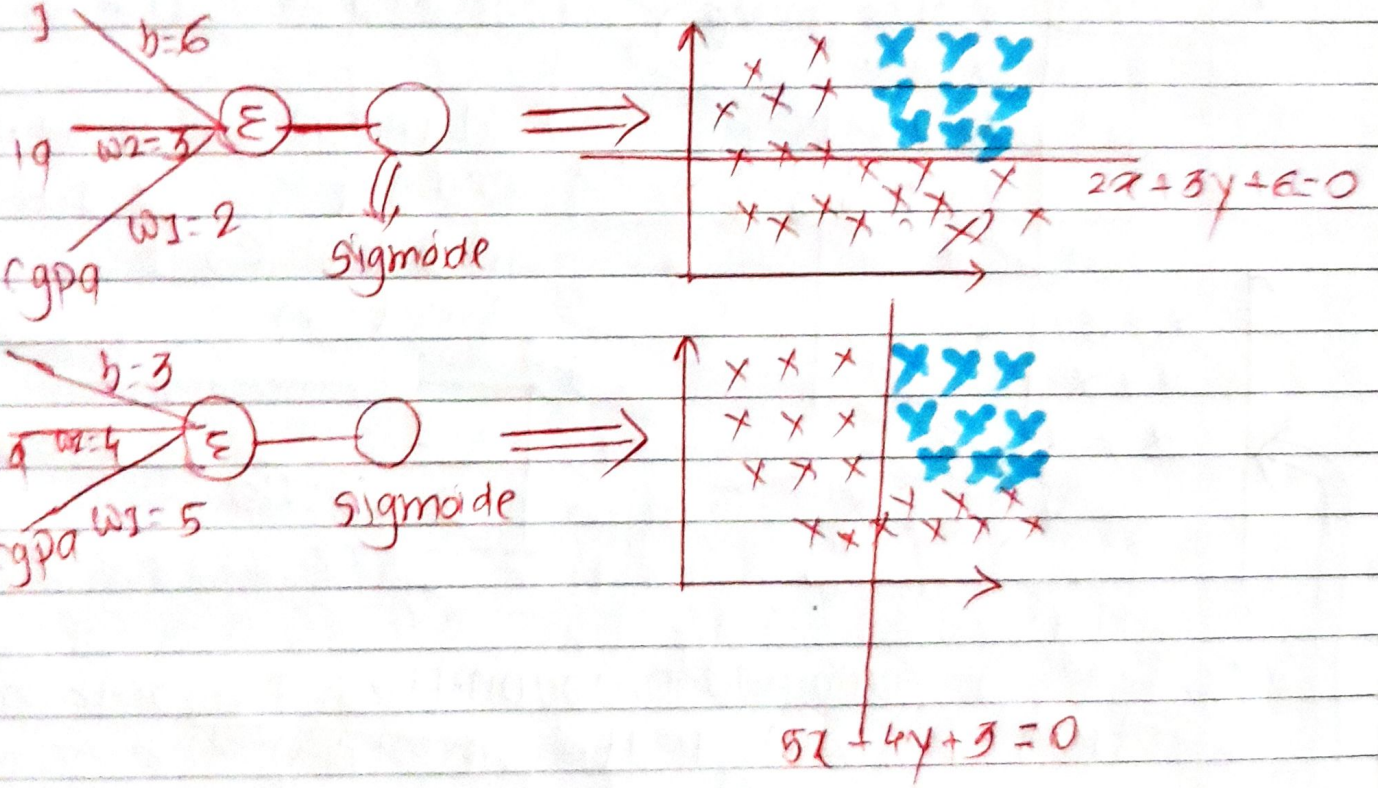
(Superimposition)

(After smoothing we get this output)

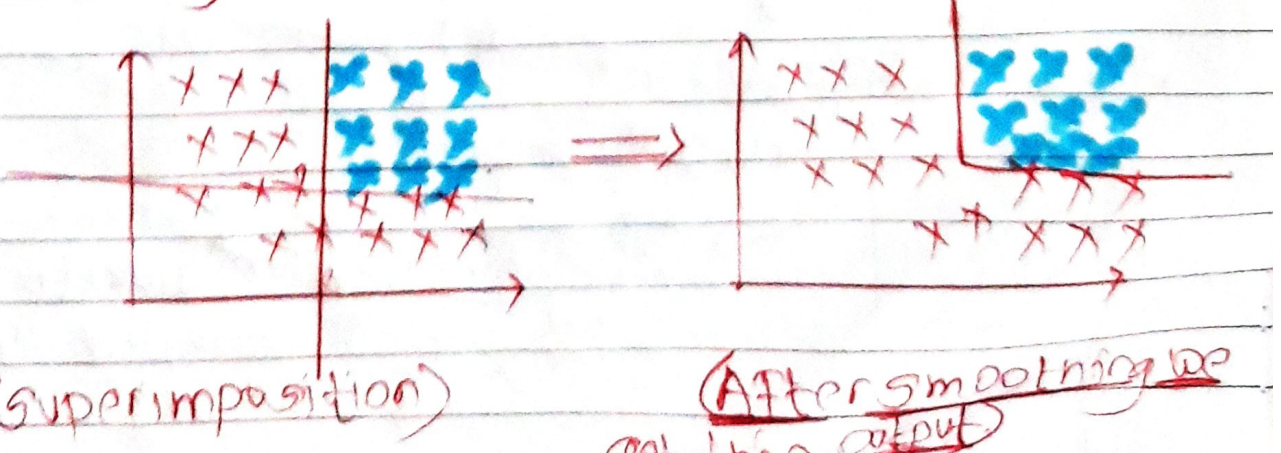
- जैसे जैसे हम line से दूर जायेंगे वैसे वैसे probabilities बढ़ेंगी, in P(getting placed) > P(not getting placed) region and opposite side probability of getting placed reduce होती है

Let's form a idea of multilayer perceptron

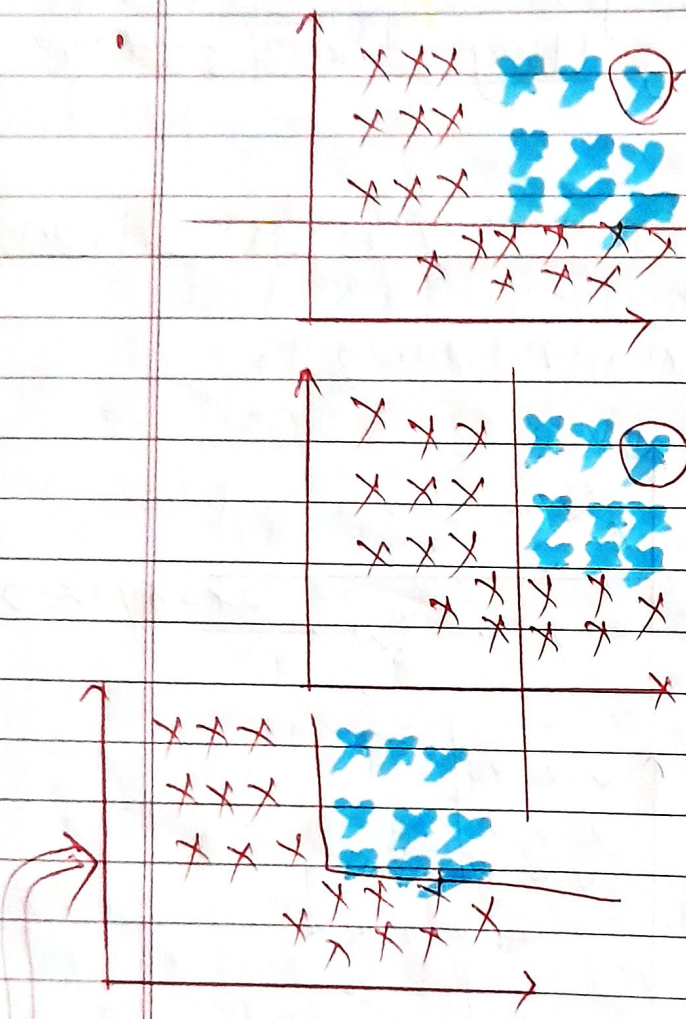
- Apply two perceptron on some dataset.



- By combining both of perceptron output we get this,



Super imposition and smoothing mathematical intuition



Probabilities of getting placed for this student according to individual perceptron

First perceptron ⇒ 0.8

Second perceptron ⇒ 0.7

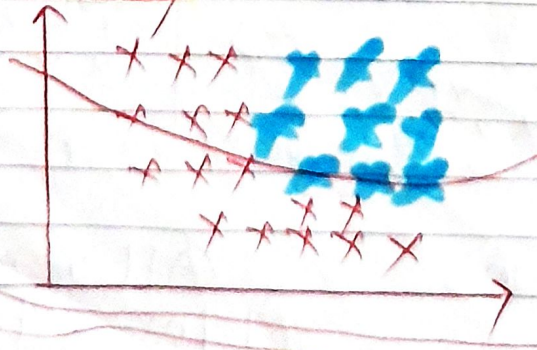
Add them and put in sigmoide function

0.8 + 0.7 = 1.5

$$\frac{1}{1+e^{-1.5}} = 0.85$$

- 0.85 is combined probabilities for both of perceptron, this will create smoother the boundry.
- Check diagram of result in back page.

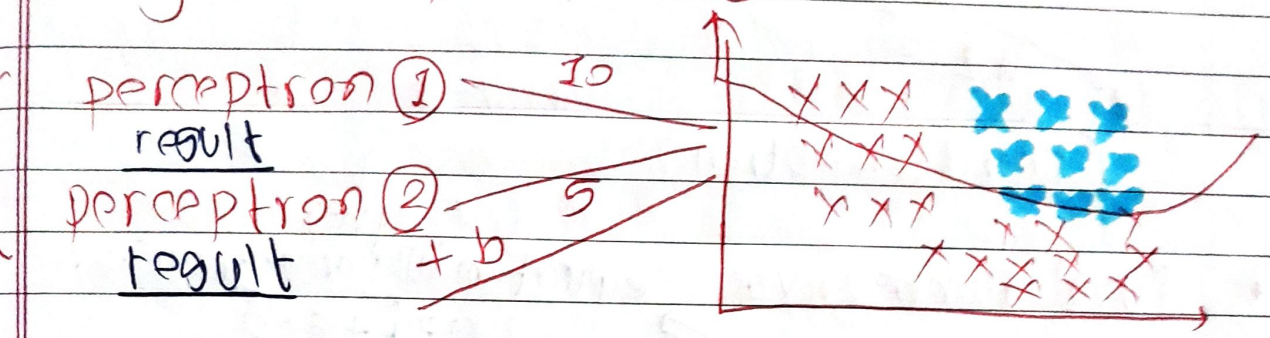
Adds Flexibility



We want this decision boundary instead of this

- This result has perception (1) import more and perception (2) less

- At the time of superimposition give more weight's to perceptron (1).



perceptron (1) output $\Rightarrow 0.7 \times 10$
perceptron (2) output $\Rightarrow 0.8 \times 5$

$$0.7 \times 10 + 0.8 + 5 = 7 + 4 = 11 = z$$

$$\text{sigmoide}(z) = \frac{1}{1+e^{-11}} =$$

- We have to do above for all points.

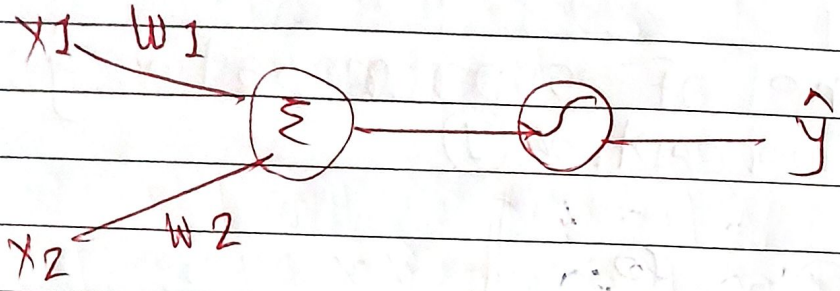
- We can add bias here too bias = 3.

now, $7 + 4 + 3 = 14$

$$\frac{1}{e^{(-14)}+1}$$

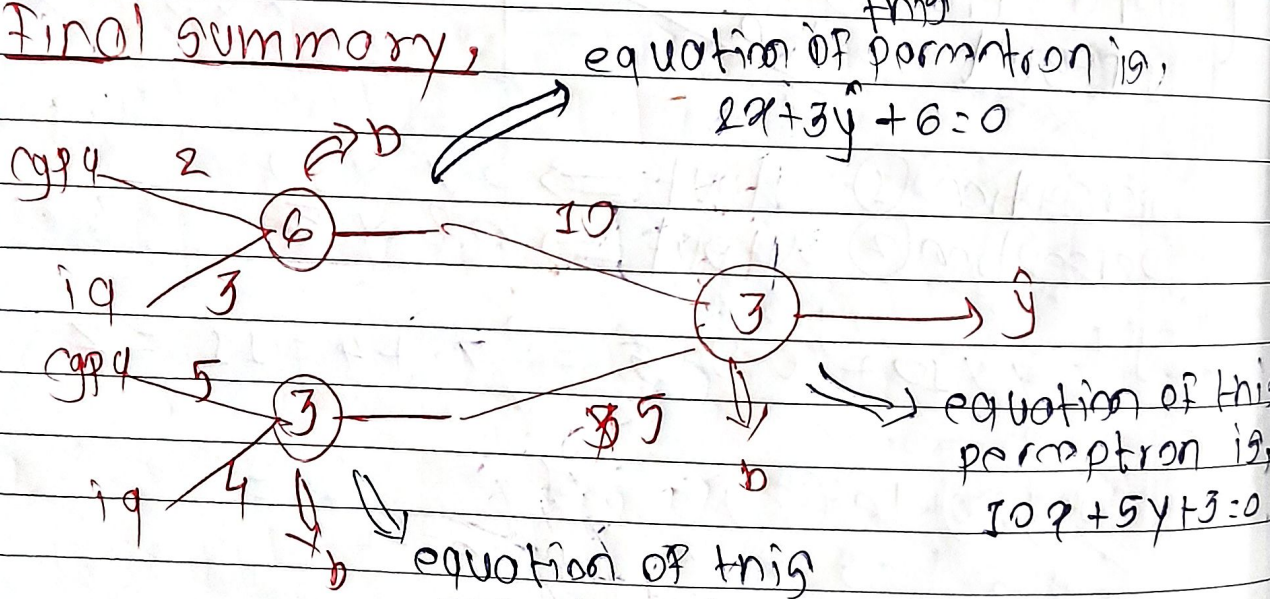
- So, this working as perceptron too, we use '3' perceptron that's why this concept.

multilayer perceptron.

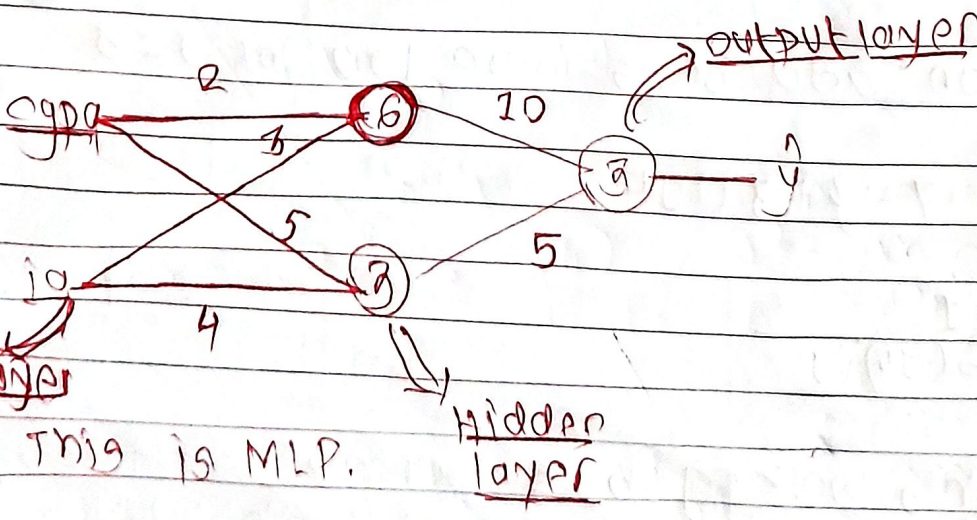


(Third perceptron).

- Final summary,



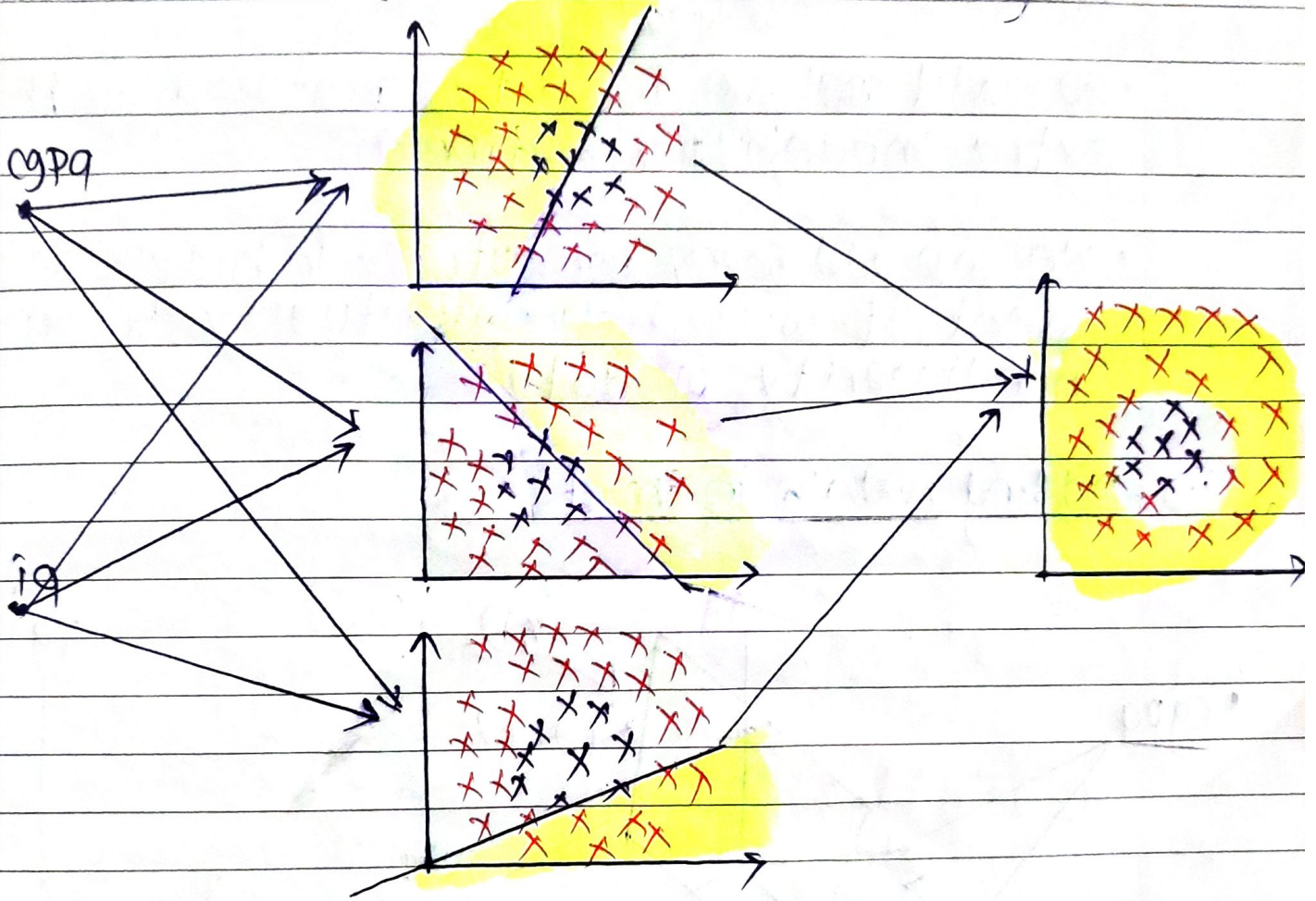
- After rearranging it,



This is MLP.

Adding layers in hidden layer (How to change a architecture)

- To have some additional flexibility.



Before that what is architecture of neural network?

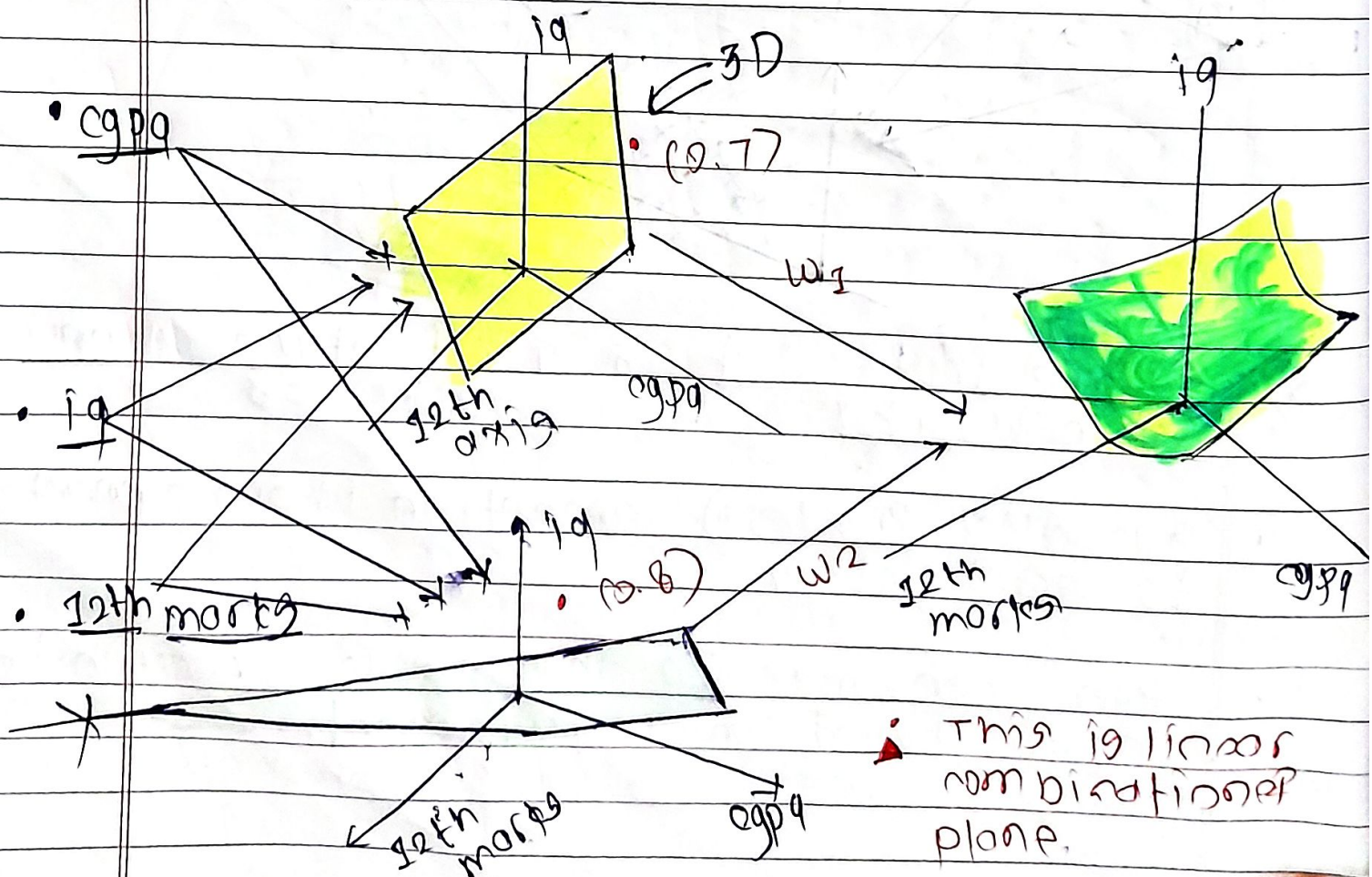
- Basically (nodes) 'perceptron' how connected to each other.
- We can make/a changes in our architecture based on requirement.

Types of changes

i) Increase number of node/perceptron in hidden layer.

- We can add many perceptrons in hidden layer as we want it will help eventually to capture non-linearity of data.

ii) Adding nodes in input



- This is linear combination plane.

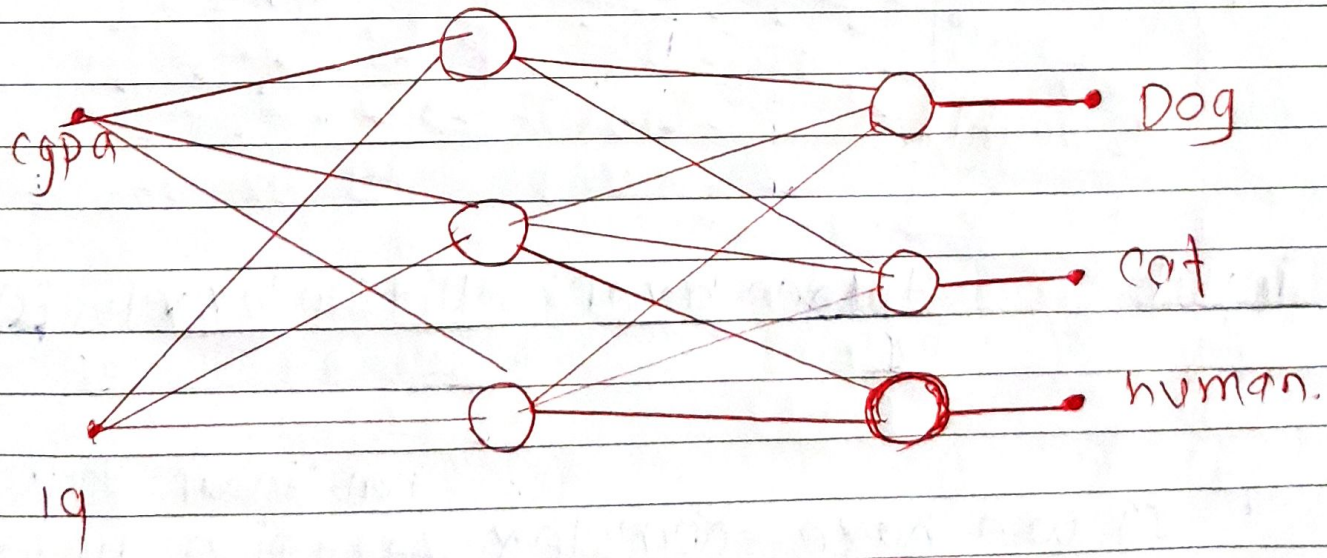
Z becomes,

$$0.7 w_1 + 0.8 w_2 = z$$

and this 'z' passes to sigmoide function for third perceptron and we get probability that point for linear combination

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

iii) **Adding nodes in output node.**



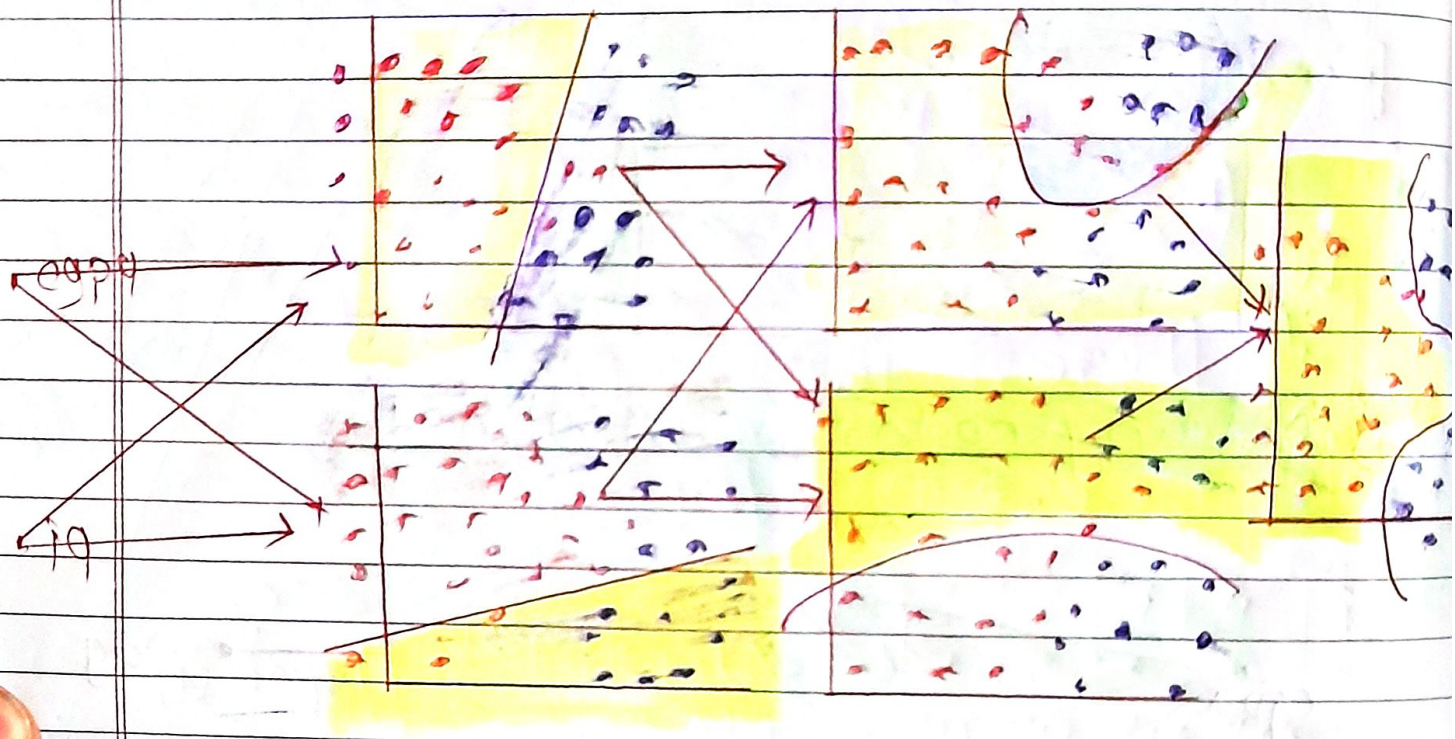
input layer | Hidden layer | Output layer

- Generally we do this if we want to do multiclass classification.

- end में जिसका भी probability ज्यादा होगा उसको हम बोलेंगे वो cat है, dog or human.

iv) Deep neural net

- Add multiple hidden layers.



- If we have complex non-linear data ~~decision~~ boundry we can differentiate by adding complex hidden layers.
- We can able to capture complex non-linearity in data.

- That's why neural network are known as universal function approximator.
- Demo in tensorflow playground.

MLP Notation

- To understand back propogation we have to learn MLP notation (We build neural network it comes with biases and weights, if we not able to denotes this notation properly then back propagation करने time confusion बहुत होता है).
- We are going to learn two things,

i) Just seeing neural network we are able to figure out how many numbers of trainable parameters included here.

Total trainable parameters } = (No. of weights and biases)

↓

कितना value हमे find करना है.

ii) How we denotes this weight and biases of this neural network.