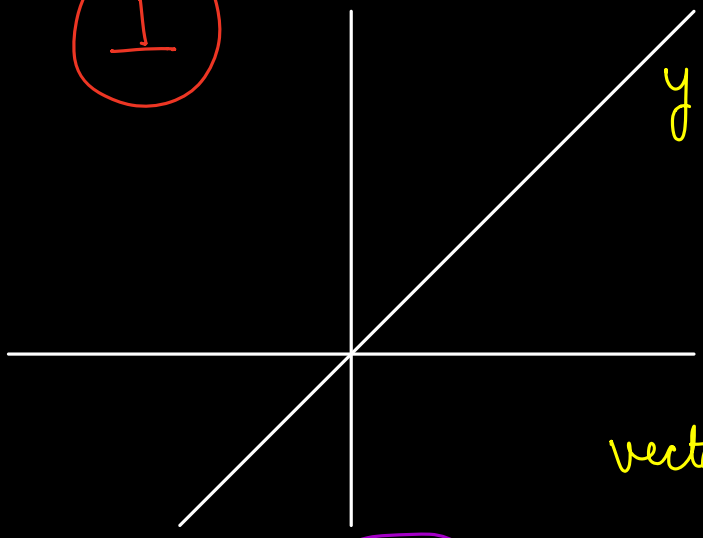
Linear equation in Matrix form

I



$y = mx + c \Rightarrow ax + by = c$

$W_1 x + W_2 y = W_0$

OR we can write it in the vector form.

$X = \begin{bmatrix} 1 \\ x \\ y \end{bmatrix}$ or $\begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$

What is Physical significance of (m) or Slope or $\frac{dy}{dx}$ or Derivate

$W = \begin{bmatrix} W_0 \\ W_1 \\ W_2 \end{bmatrix}$

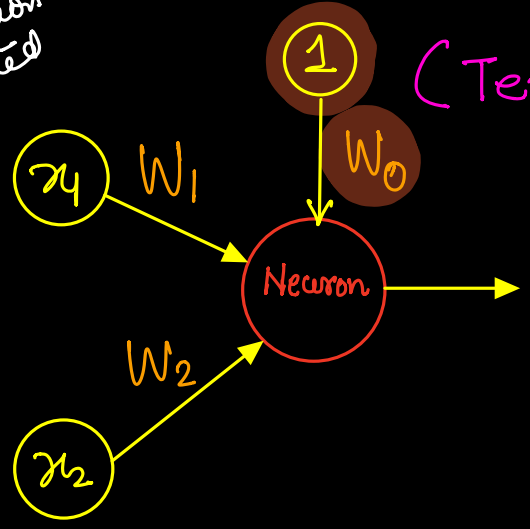
$\therefore \boxed{W^T X = 0}$

II

Given a line say $(y = mx + c)$ & a point (x_1, y_1) How to known its side.

$\boxed{y_1 - mx_1 - c \geq 0}$ OR $\boxed{y_1 - mx_1 - c < 0}$

This computation can is abstracted as a neuron.



This neuron just works as an aggregator and do (2) things

(a) Compute $W_0 + W_1 x_1 + W_2 x_2$

(b) fires if it is ≥ 0

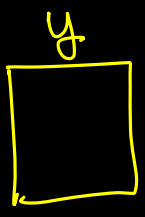
Derivative

$m = \frac{dy}{dx}$ which is rate of change of (y) wrt (x)

$\Downarrow\Downarrow\Downarrow$

How much is Δy when $\Delta x = 1$

Just think (x) as a knob and we need to know how much it can effect (y).



[its sensitivity wrt to the value of (y)]

III What is a function?

It is just a mapping b/w input & o/p.

$y = f(x)$ (1D function)

Plotted in 2D space

Can only be differentiated in one direction

$\left[\frac{dy}{dx}\right]$

(Multivariate Calculus)

$y = f(x_1, x_2)$

Plotted in 3D space.

Can be differentiated in (2) directions

$\left[\frac{dy}{dx_1} \text{ \& } \frac{dy}{dx_2}\right]$

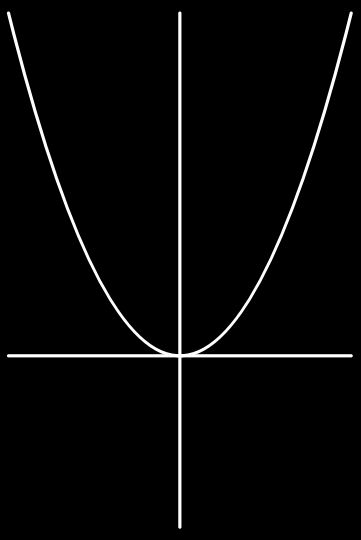
y differentiate wrt vector $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

But why derivatives are essentially required.

IV

Given any function $f(x)$ how can you maximize or minimize. [Optimization]

Say $y = f(x) = x^2$ (Quadratic fn)



$$\frac{dy}{dx} = 2x \quad \therefore 2x = 0 \Rightarrow x = 0$$

where we can get extremas.

If $y = (x+1)^2$ $\quad \frac{dy}{dx} = 2(x+1) = 0 \quad (x = -1)$

(How to know maxima OR minima)

$ax + by + cz = d$ (This is a plane in 3D)

$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 = 0$ (What is this)

$$\sum_{i=0}^{n} w_i x_i = 0 \quad \text{(what is this?)} \quad (x_0 = 1)$$

(Hyper-plane) Multidimensional linear function.

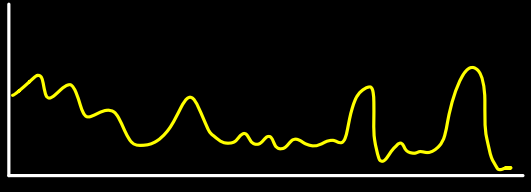
(V) Given a multidimensional fn say (F) defined in $\mathbb{R}^N$ How can we minimize it. This (F) can be a loss function or objective function. (Closed-form)

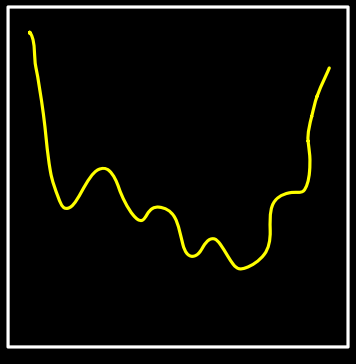
In some (N) dimensional (say $w_0, w_1, w_2 - - - w_{N-1}$) Parametric space assume that some loss function (F) is defined. We require :

$$W^* = \arg\min_{W} (F)$$

Just differentiate (F) wrt W $\Rightarrow \frac{dF}{dW}$

This can be done easily if (F) can be represented as some Close form formula (as discussed above)





But most of the functions are not well defined as some formula and are represented in a discrete manner.

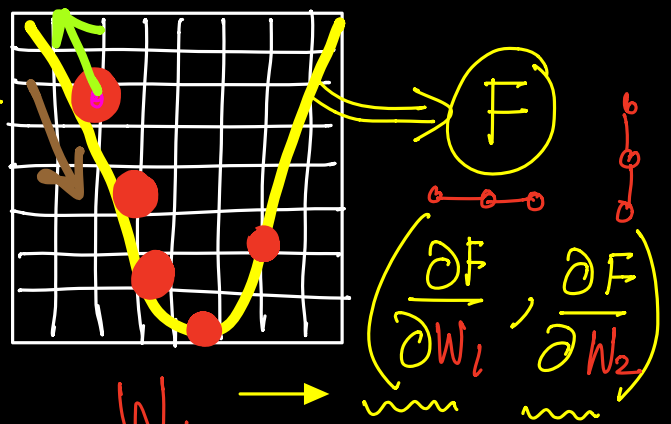
$f(x_1), f(x_2), f(x_3)$ - - - - -

OR $\forall_{x_1, x_2, x_3} f(x_1, x_2, x_3)$

Discrete functions

VI

Now in such a scenario how can be optimize a an objective function.



In discrete space we have an iterative solution.

$$W_1^{New} = W_1^{Old} - \eta \frac{\partial F}{\partial W_1}$$

$$W_2 = W_2 - \eta \frac{\partial F}{\partial W_2}$$

Parametric Search for optimization. [Update till Convergence]

Gradient defined in a neighbour. (finite differencing)

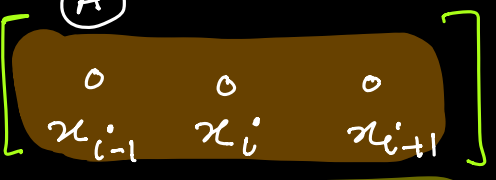
Gradient decent algorithm for optimization (used as optimizer)

SGD, ADAM, AdaGrad e.t.c.

Analytically computing gradient:

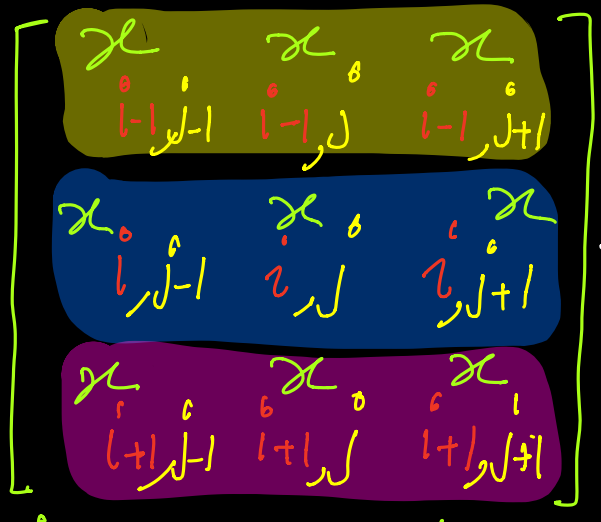
$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

(A)



for (x_i), gradient in horizontal direction can be: $x_{i+1} - x_{i-1}$

(B)



for $(x_{i,j})$ gradient in horizontal:

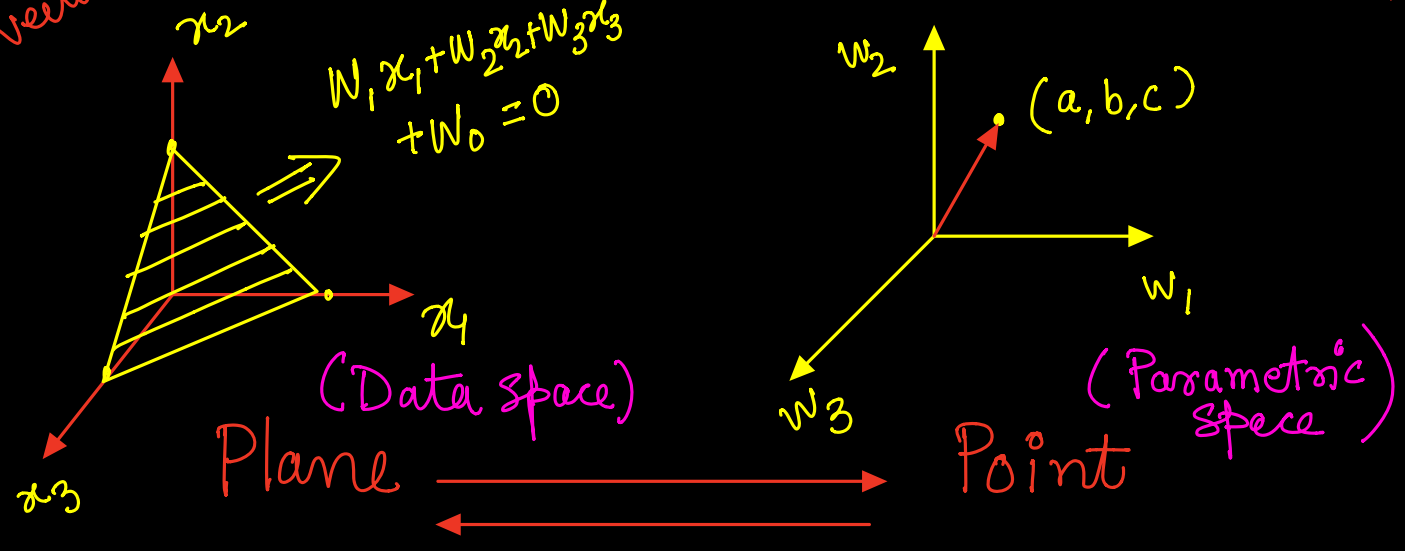
$x_{i,j+1} - x_{i,j-1}$ OR averaging all (3) Rows

This is gradient wrt a neighbourhood.

VII Hence in that high dimensional space (N-dim)

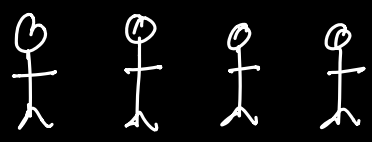
- (*) These are (N) directions to move on.
- (*) we are in search of optimal set of parameters (W^*) that can minimize the objective fn (F).
- (*) we need to use Gradient decent iterative algorithm to optimize over (F) in $\mathbb{R}^N$.
  - → Initialize (W) vector $\in \mathbb{R}^N$, randomly.
  - → Update $\forall_i \; W_i = W_i^{old} + \Delta W_i$
    where $\{\Delta W_i = -\eta \frac{\partial F}{\partial W_i}\}$
  - → Each [N] vector $\in \mathbb{R}^N$, and can be seen as a h-plane in the space of (X).

$WX = 0$ is a h-plane in vector form in $\mathbb{R}^N$

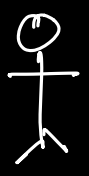


VIII

Problem of binary classification



(1000 persons have applied for
(a) Join by research group
(b) Join my cricket team.



Name
Mobile
Qualification
Gender

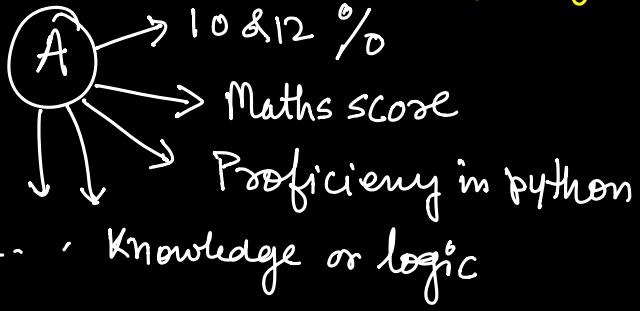
* (How many attributes to be considered)? as well as what?

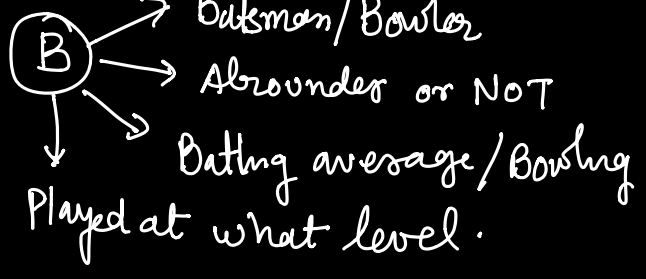
* Each actual data may have infinite amount of information.
like (father's Name, Village, Uncle, family, friend, 3D gesture, face structure - - - - - - ...)

Central question

* During the interview what questions we need to ask.

* What is the weightage of that question.





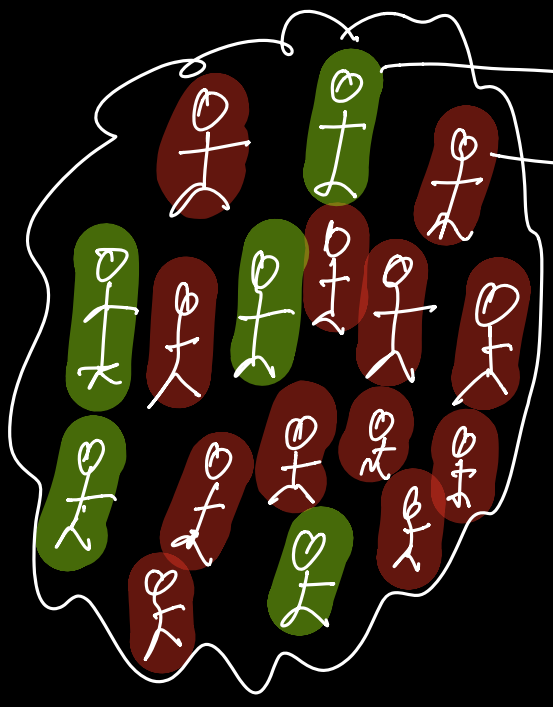
Feature selection

* One can observe that the questions that we need to ask are depending upon the Objective or Task in hand (Problem Statement)

Question and their importance keep on changing with respect to the problem in hand [Feature Engg]

Now as an interviewer, I will ask questions to all the applicants. Get their answers (features) and accept/reject (Based on some Criterion)

[Classification decision]



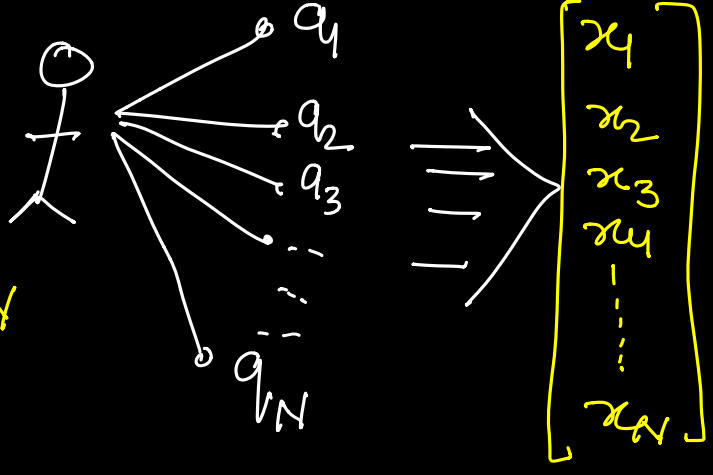
→ Selected

→ Rejected

(Out of 1000 100 selected & 900 got rejected)

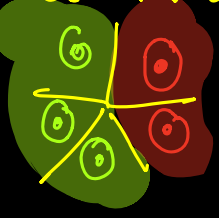
(*) What we need to learn : The classification decision rule, given the questions.

(*) Assuming we ask (N) questions to each applicant:



☺ Basically each data got projected onto $\mathbb{R}^N$ Space: data to an N-dim vector

☺ In that N-d space ($\mathbb{R}^N$) it will be seen as a point.

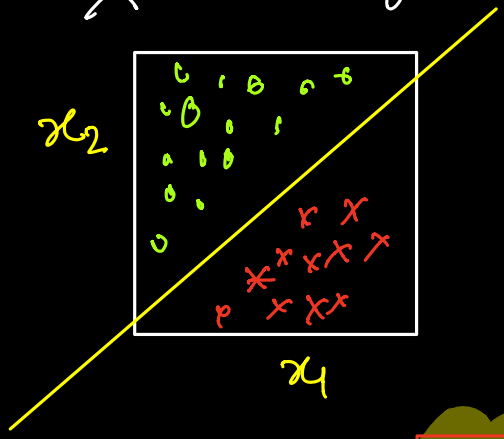


Typical (N) values are 128/256 & much bigger.

* We need to ask relevant questions, so as to project data points into $(\mathbb{R}^N)$ in such a way that in that space, accepted & rejected Candidates are well apart.

Feature extraction

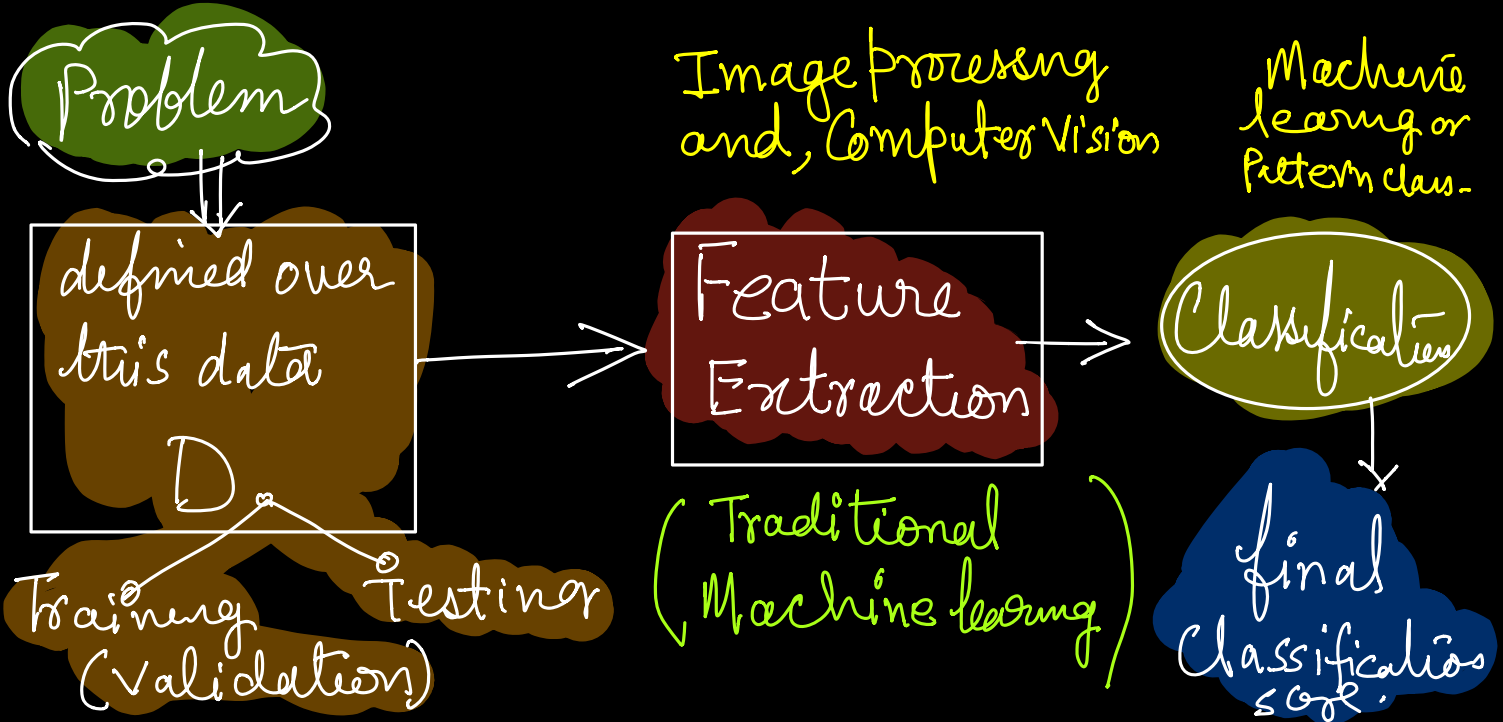
* Assuming we have asked only (2) questions
  → data got projected in (2D) space.



Classifier need to learn this seperating line or H-Plane Called Decision boundary.

ML to DL

The full flow of a classification system can be



Similar to the above mentioned functional Optimization, we need to search for the best h-plane in the feature space $X = \{x_1, x_2 - - - x_N\} \in \mathbb{R}^N$

Which will be

$\underbrace{W_0}_{\text{Bias}}, W_1, W_2 - - - - W_N \}$ Coefficients of the h-plane.

But what is going to be the function that we need to optimize $\Rightarrow$ loss fn

- MSE or MAE (for Regression)
- Cross Entropy (Classification)

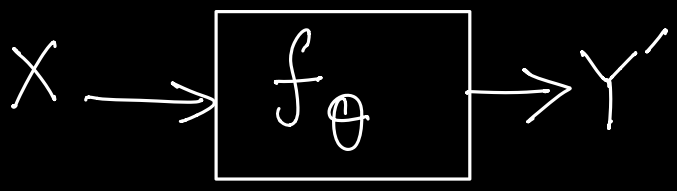
Supervised Learning:

These loss fn, for each given i/p data (X) assumes a correct o/p (Y) say ground truth. Using the learned logic, (X) will be converted into the predicted o/p (Y'). Loss functions need to estimate the dissimilarity/deviation b/w

$Y \& Y' \Rightarrow \{LF(Y, Y')\}$

In case of regression Y & Y' may be a single value

for classification Y & Y' $\Rightarrow$ Probability scores for all the classes.



data have the corresponding labels $\{ X_1 \to Y_1,\ X_2 \to Y_2,\ \ldots,\ X_K \to Y_K \}$

(A) **Regression** need to learn (θ) (Set of parameters) such that

$f_\theta(X_i) \equiv Y_i'$ need to be as close as possible to Y_i. Loss fns can be:

L_2 Norm (a) $MSE(Y_i, Y_i') = \frac{1}{2}(Y_i - Y_i')^2$

L_1 Norm (b) $MAE(Y_i, Y_i') = |Y_i - Y_i'|$

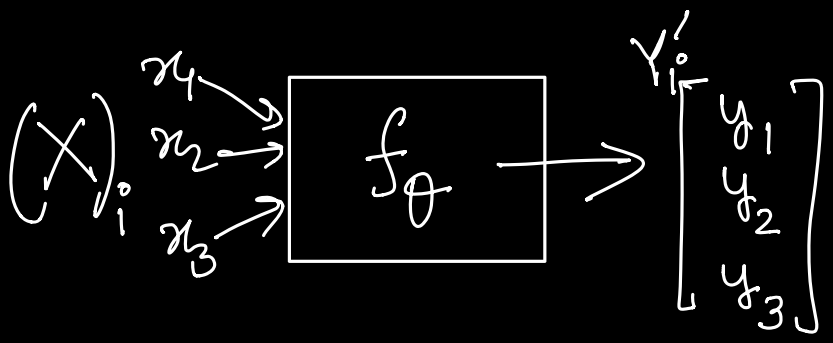
(B) **Classification**: Y_i's are represented as 1-Hot encoding.

if (10) class classification then

each $Y_i \Rightarrow$ 10-dim (binary vector)

$$\begin{bmatrix} \text{Cat} \\ \text{dog} \\ \text{Car} \\ \vdots \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow$$

implies Y_i belongs to Cat (only one place is set)



3 class problem

$\Rightarrow Y_i'$ is 3-Dim

and Y_i is also 3Dim

$\forall_i$ X_i we have their corresponding labels (Y_i) (3D)

Loss fn : Cross entropy (Y_i, Y_i')

$\Rightarrow$ Y_i | y_1 | y_2 | y_3 |

$\Rightarrow \sum_{i=1}^{3} - y_i \log(y_i')$

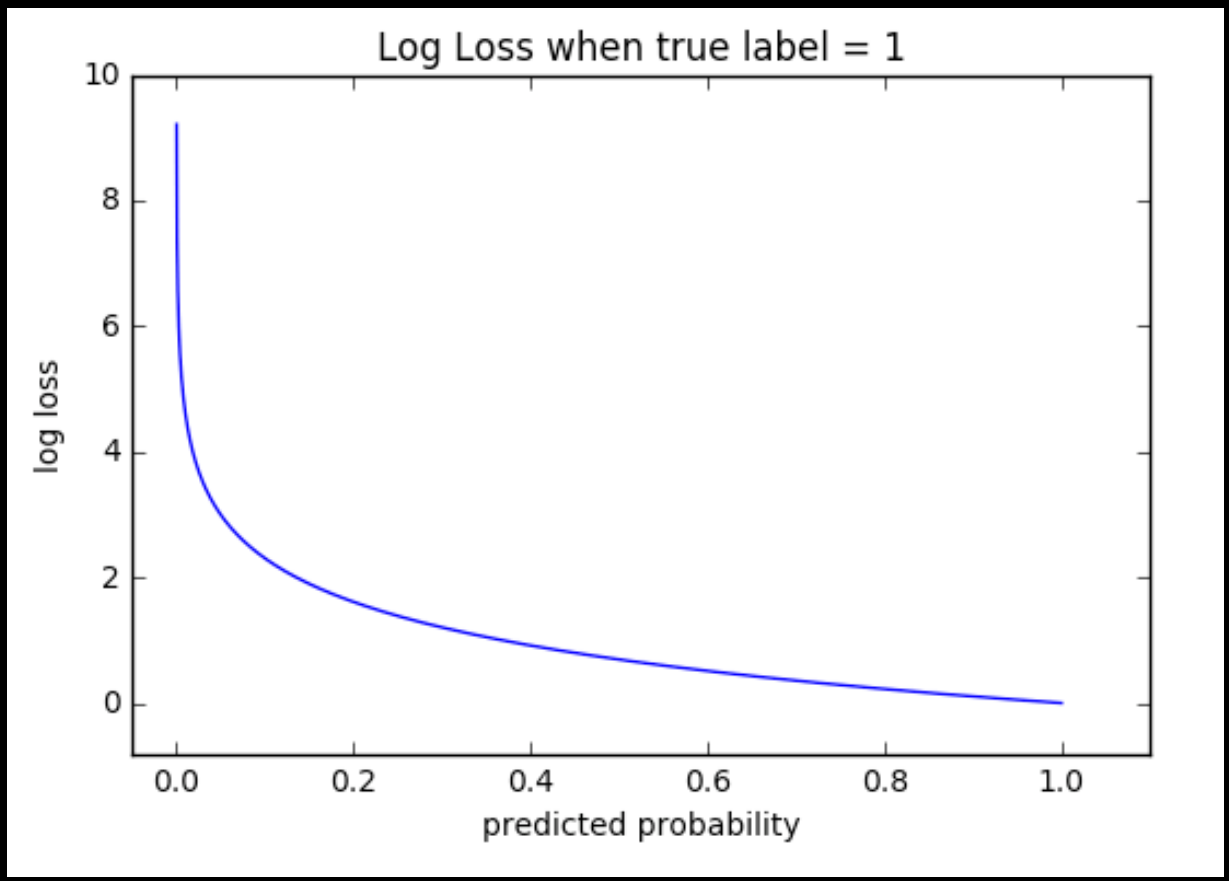
Y_i' | y_1' | y_2' | y_3' |

This is $\Rightarrow -(y_1 \log(y_1') + y_2 \log(y_2') + y_3 \log(y_3'))$ ....

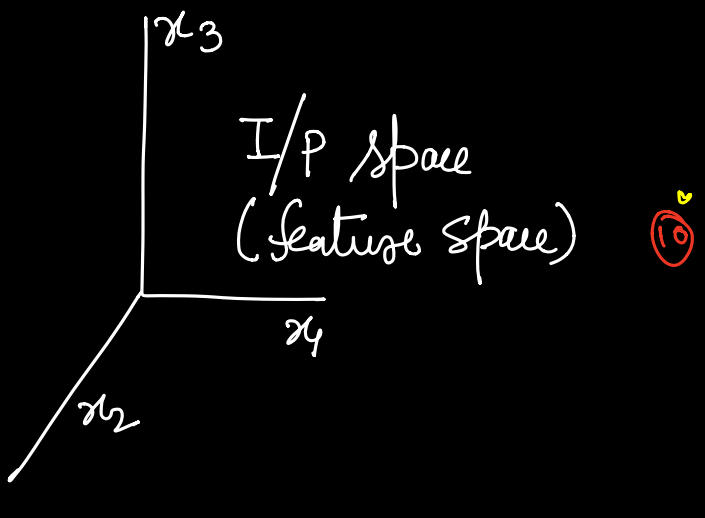
y	p
$1-y$	$1-p$

Binary cross entropy

$-(y \log p + (1-y)\log(1-p))$



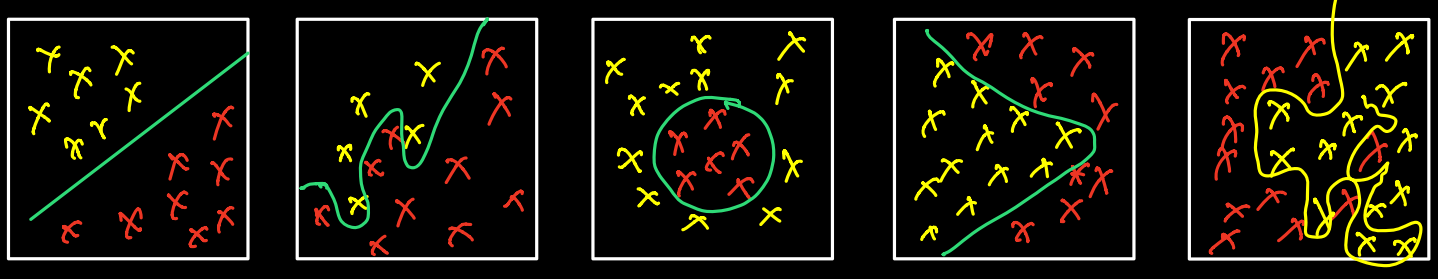
(*) So, now we are having 2 spaces



(*) Each point here is a plane in feature space with some corresponding loss associated to it.

(*) we need to explore the parametric space inorder to optimize the Cost fn efficiently (This is done by applying GD or SGD)

(*) One neuron can be used to learn one such h-plane if our data is linearly seperable. But mostly data is not linearly seperable. Then how to learn non-linear boundaries.



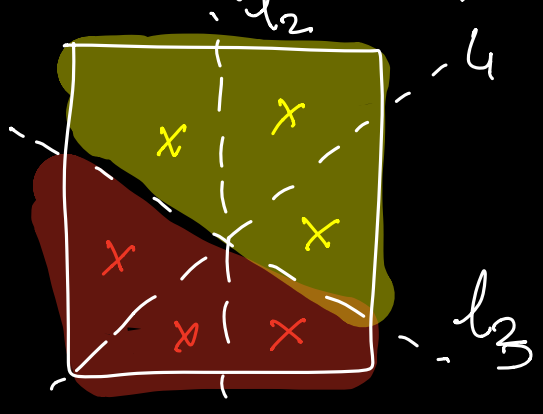
We require more neurons, may be in multiple layers.

Intuition of Neural Networks

(*) Neural networks can be gaints and may have Millions of neurons & thousands of layers.

(*) Essencially they are partitioning the feature space with an objective of minimizing mis-classifications.

Basically Splitting the data



→ one can start from some very bad h-plane.

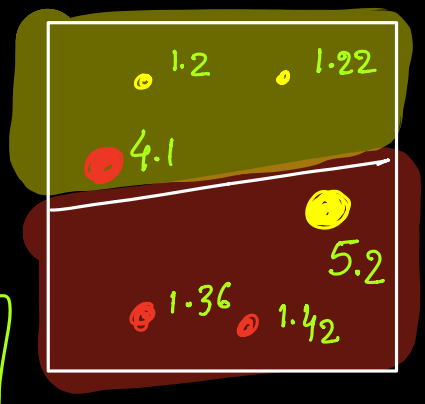
→ Need to update it parameter so as to minimize the objective fn (mis-classification)

→ We can utilize SGD & GD iteratively.

But there is an issue ⇒ Loss fn need (Here it is discrete) to be differentiable

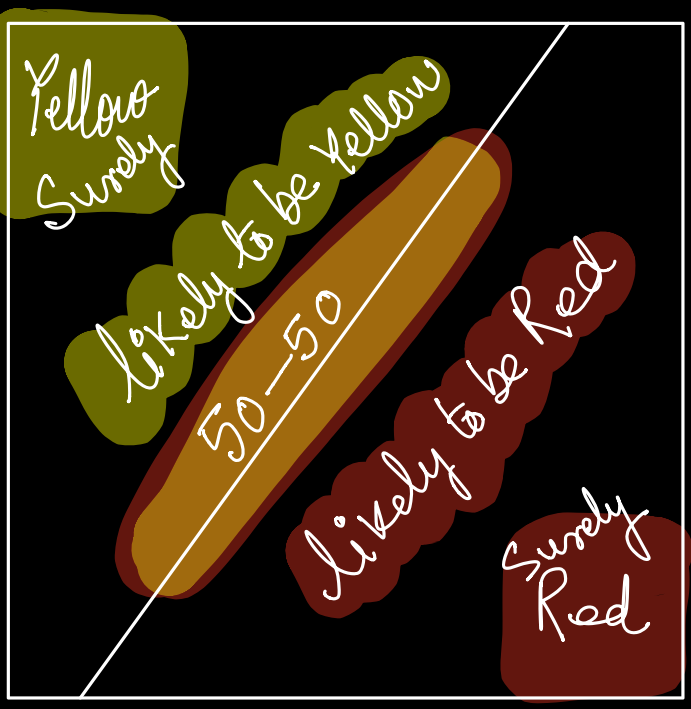
(*) We can use logistic loss fn ?

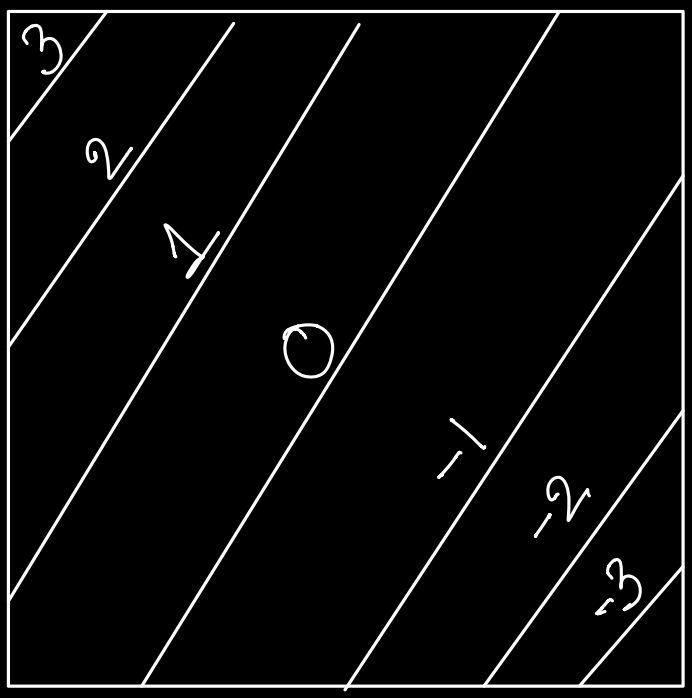
→ Each and every data is Penalized. Correctly classified then less penalty & mis-classified then more penalty



Error ⇒ 1.2 + 1.222 + 4.1 + 5.2 + 1.36 + 1.42 [Continuous]

Now the question is how to get such loss fn?

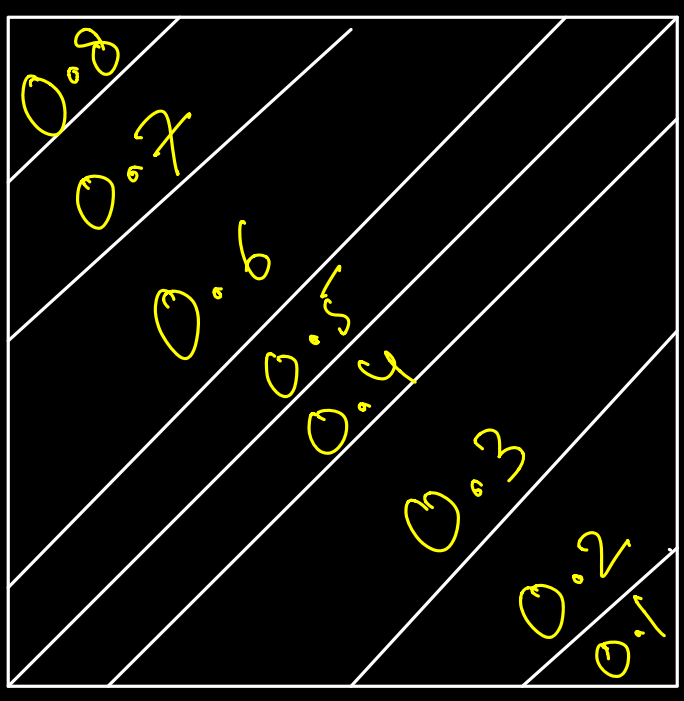


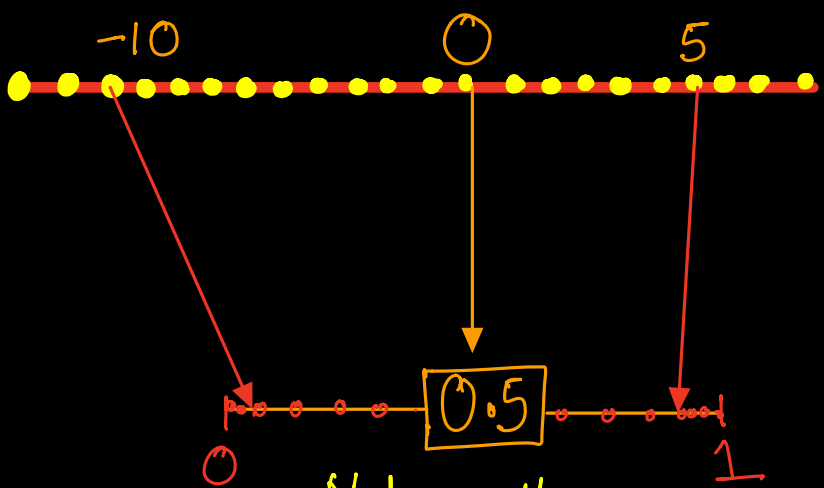


Yellowness

It can be done using Sigmoid fn :

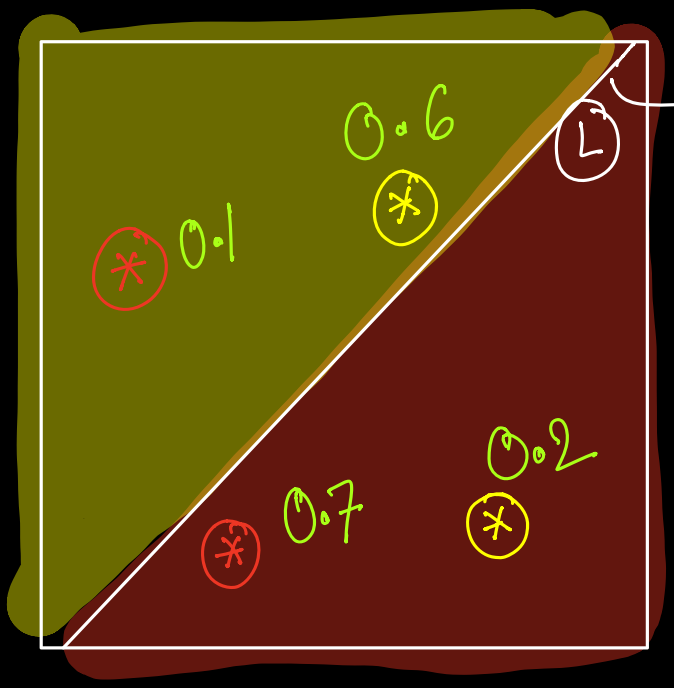
$$S(x) = \frac{1}{1+e^{-x}}$$





Not equally spaced ⟹ Basically a non-linear mapping

They are called Activation function, (Nonlinear) like, Sigmoid, Tanh, ReLU, LReLU, PReLU ----.



Let us Consider this Example.

→ Decision boundary (L) separating Yellow & Red Region.

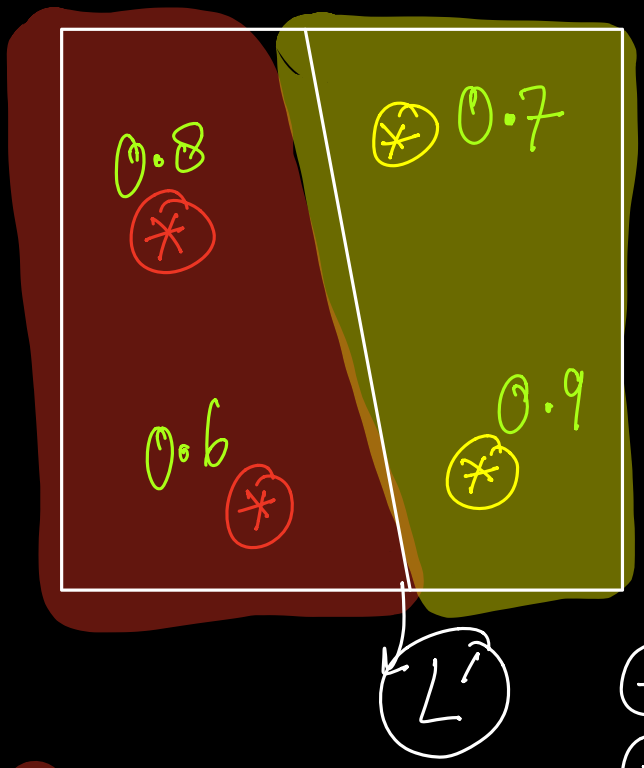
* 2 Classes Yellow / Red
* Shown values are the probability that they belong to the Correct class wrt given decision boundary (L).

∴ Joint probability that all points got Correctly classified by (L) = $0.6 * 0.2 * 0.1 \times 0.7$

$\Rightarrow 0.0084$

(This can be termed as likelihood)

∴ Likelihood can be used as a gain fn & need to be maximized.



for above shown boundary (L)

$0.6 * 0.2 * 0.1 \times 0.7$

$= 0.0084$

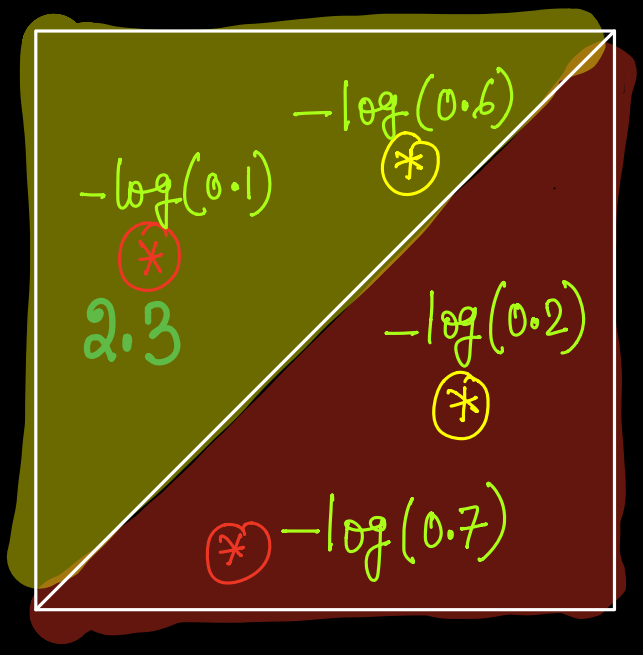
(maximized)

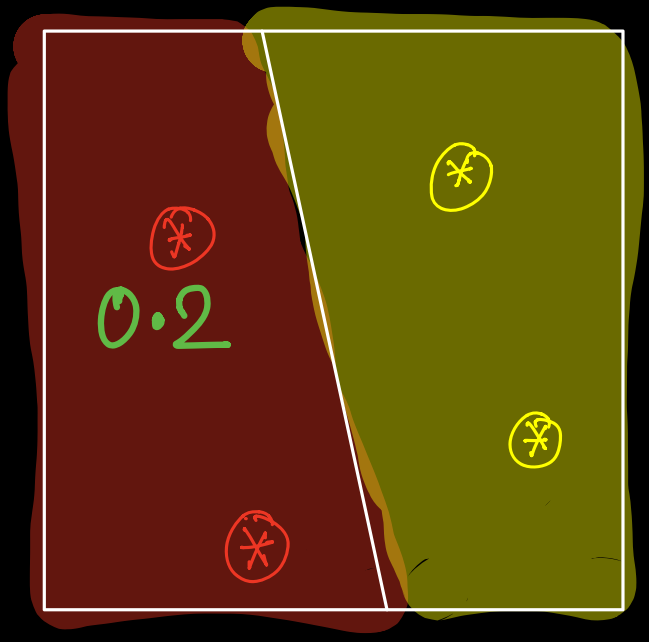
for (L')

$0.7 * 0.9 * 0.8 * 0.6 = 0.3024$

* Here issue is it is multiplication
* The values are very small.

Multiplication → Addition & value scaling.





$0.6 * 0.2 * 0.1 * 0.7 = (0.0084)$

$0.7 * 0.9 * 0.8 * 0.6 = 0.3024$

(Maximization of likelihood)

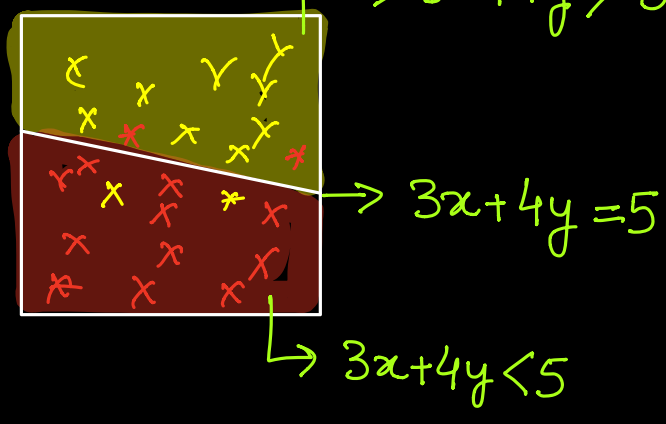
$-\log(0.6) - \log(0.2) - \log(0.1) - \log(0.7) = \boxed{4.8}$

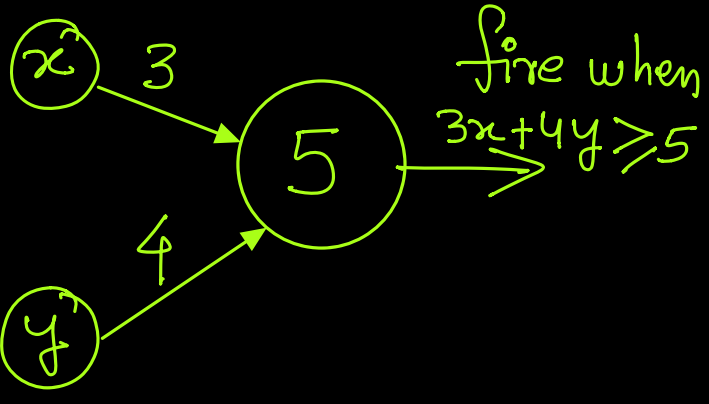
(2.3)

$-\log(0.7) - \log(0.9) - \log(0.8) - \log(0.6) = \boxed{1.2}$

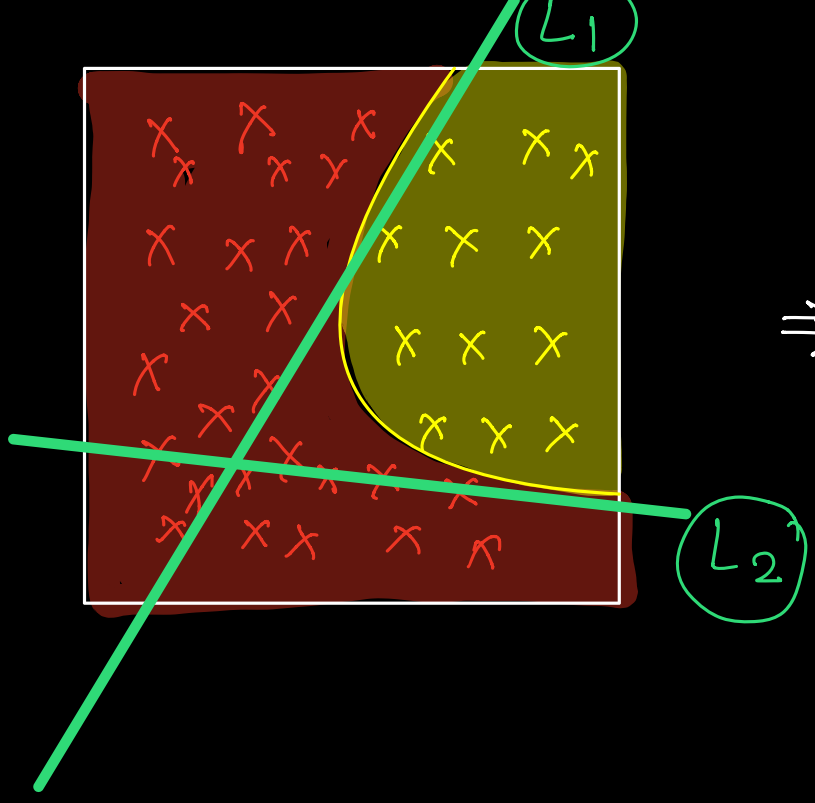
(0.2)

log likelihood minimization $\Rightarrow$ Optimize & get the linear boundary.





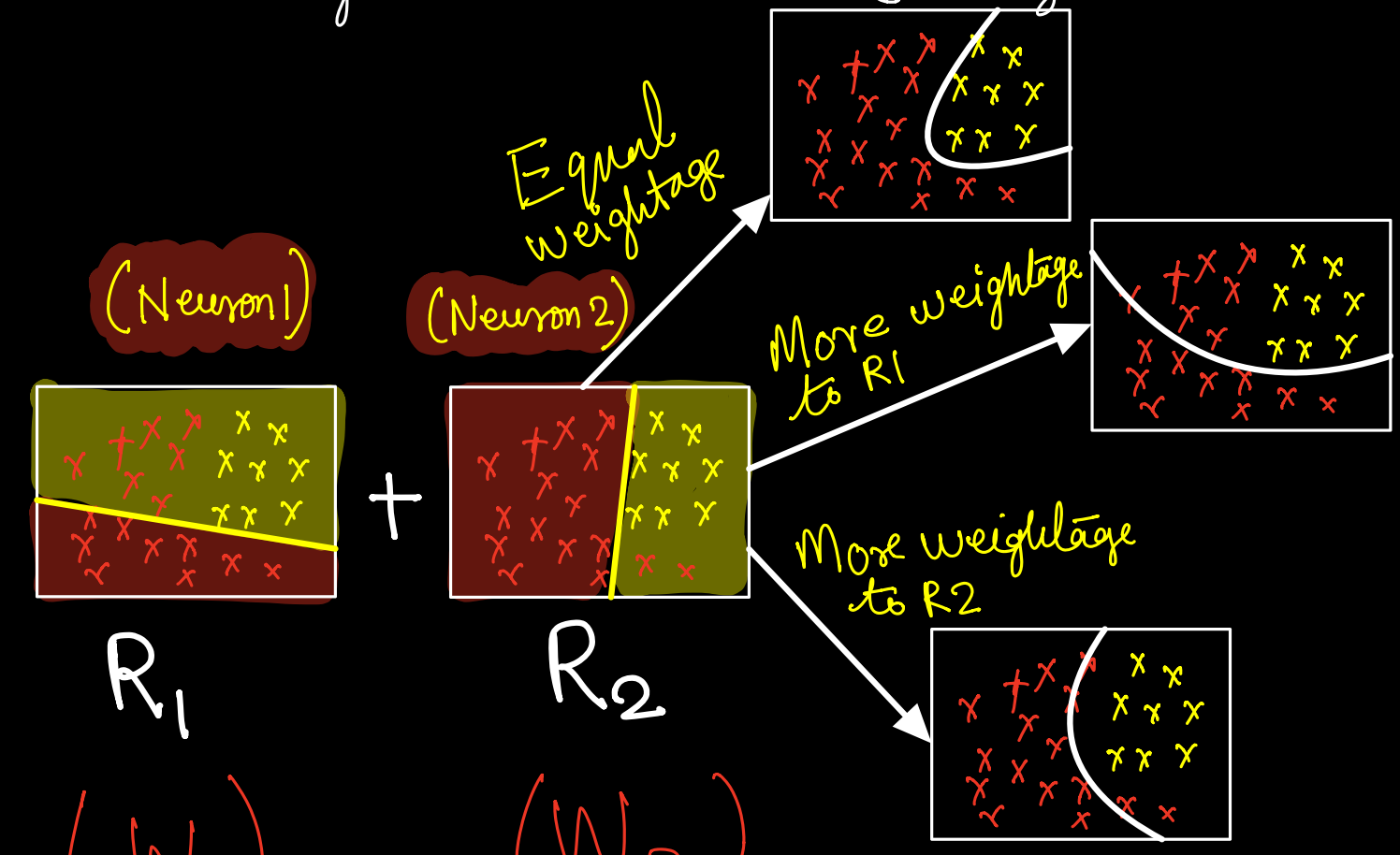
Single neuron can learn decision boundary if data is linearly seperable.



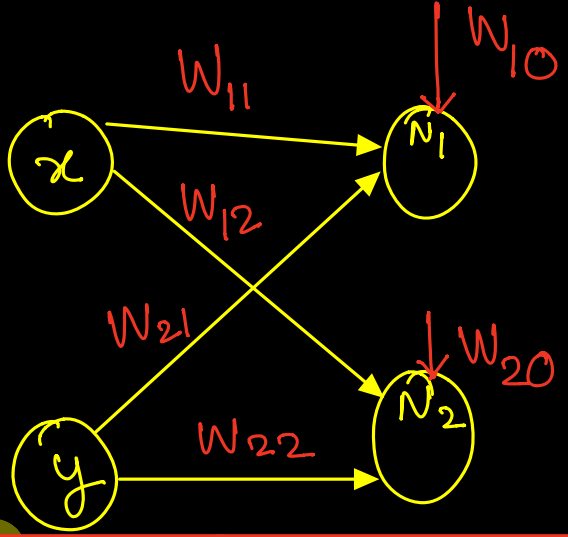
How to learn the non linearity?

⇒ we need to learn the non-linear boundaries in a modular fashion & hierarchically. (using a layered N/w)

First we need to understand Region Combination (weighted)



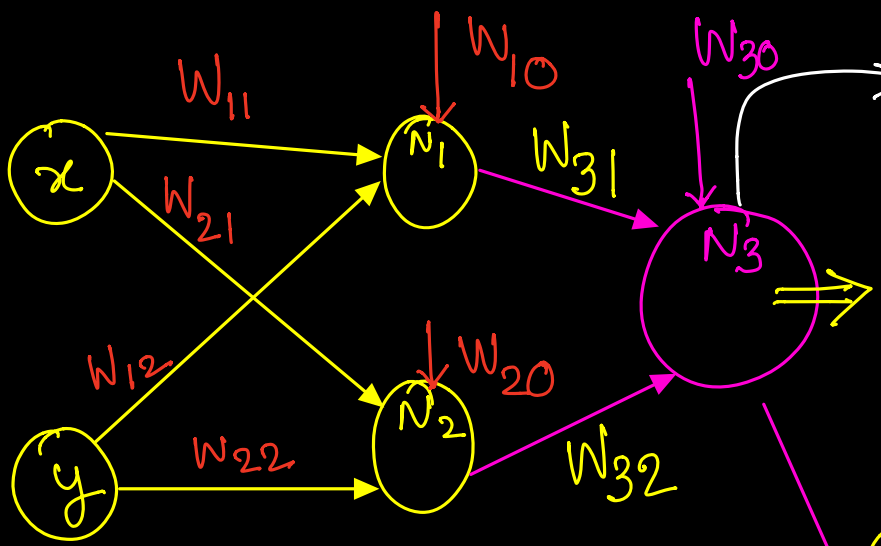
these weights also need to be learned



⇝ This neuron is learning one h-plane (learning Region 1)

⇝ This neuron is learning another h-plane. (learning Region 2)

$W_{ij} \Rightarrow i^{th}$ neuron, j^{th} input



This neuron is learning How to merge 2 regions, (Basically 2 decision merging)

(*) N_3 not seeing the actual i/p.

(*) Objectives of N_1 & N_2 (?)

neuron-1

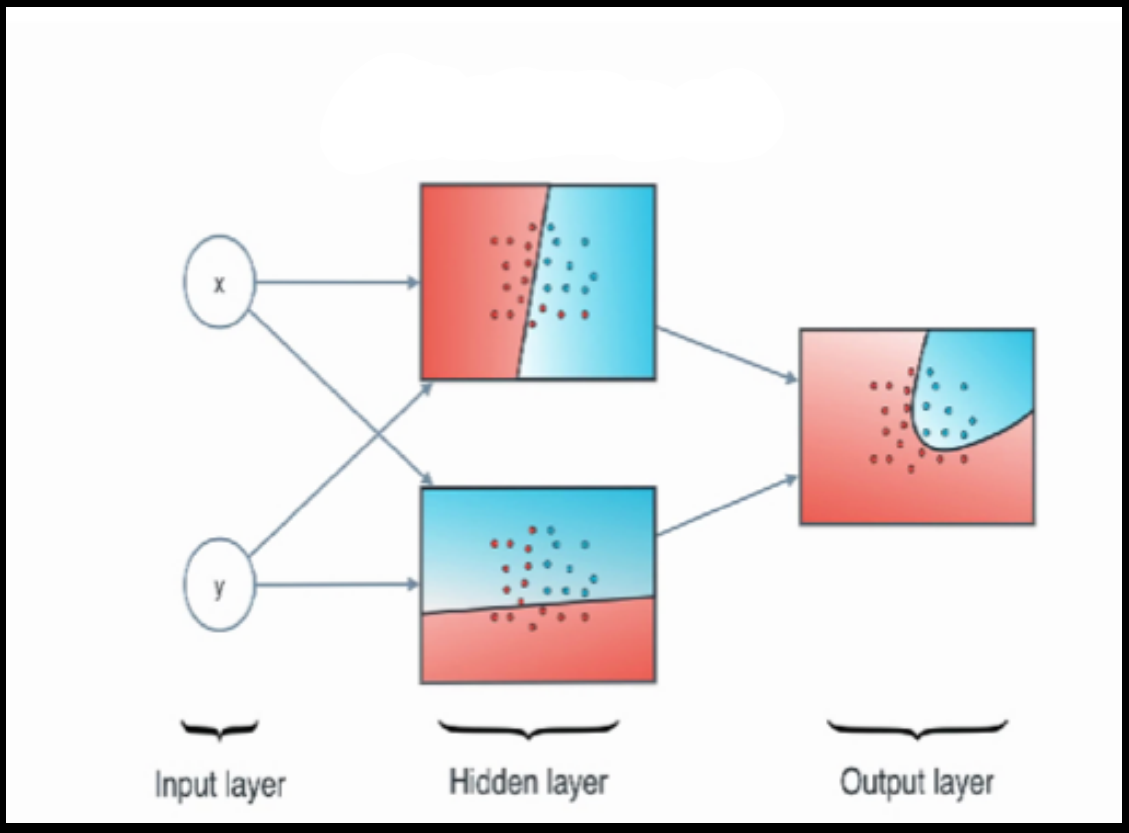
neuron-2

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} W_{10} \\ W_{20} \end{bmatrix}$$

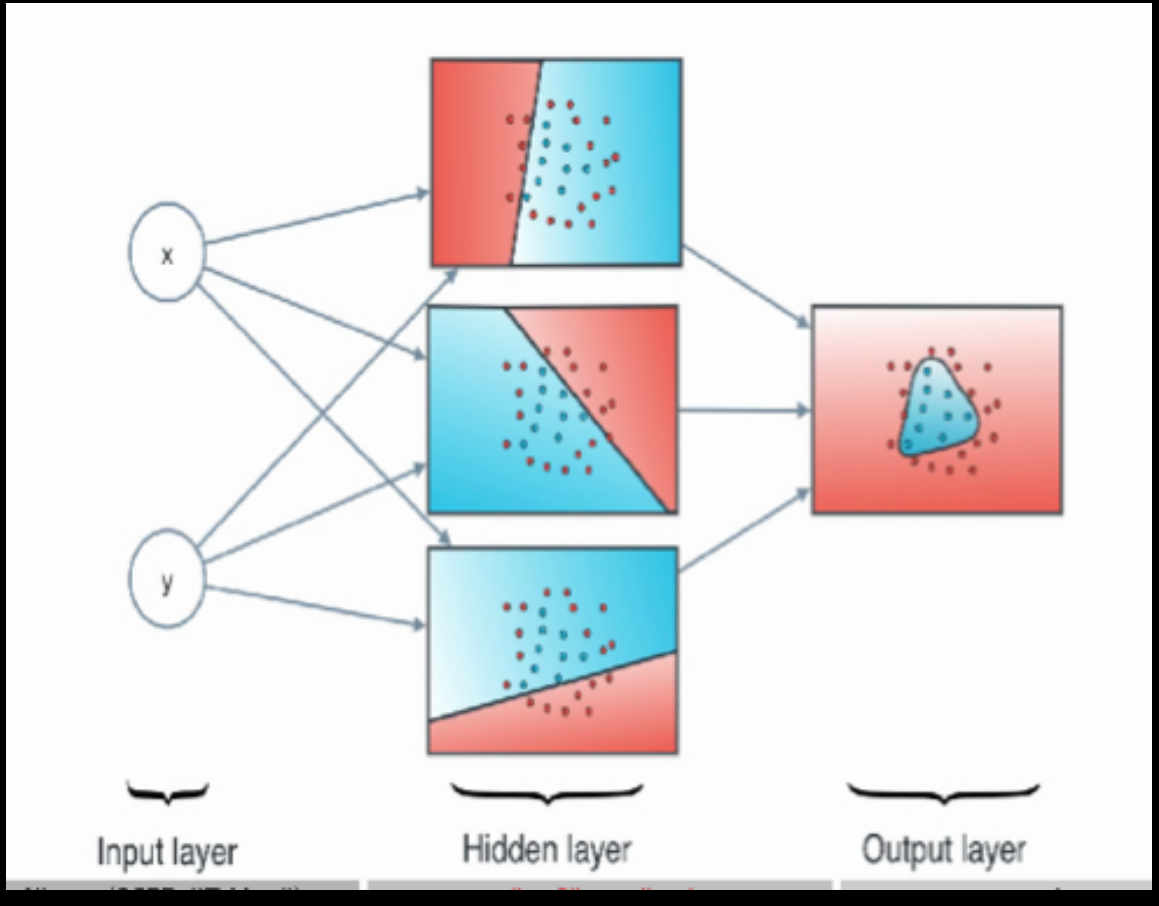
$$\Rightarrow \begin{bmatrix} W_{11}x + W_{12}y + W_{10} \\ W_{21}x + W_{22}y + W_{20} \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

$$\begin{bmatrix} W_{31} & W_{32} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

$$\equiv W_{31} I_1 + W_{32} I_2 + W_{30}$$

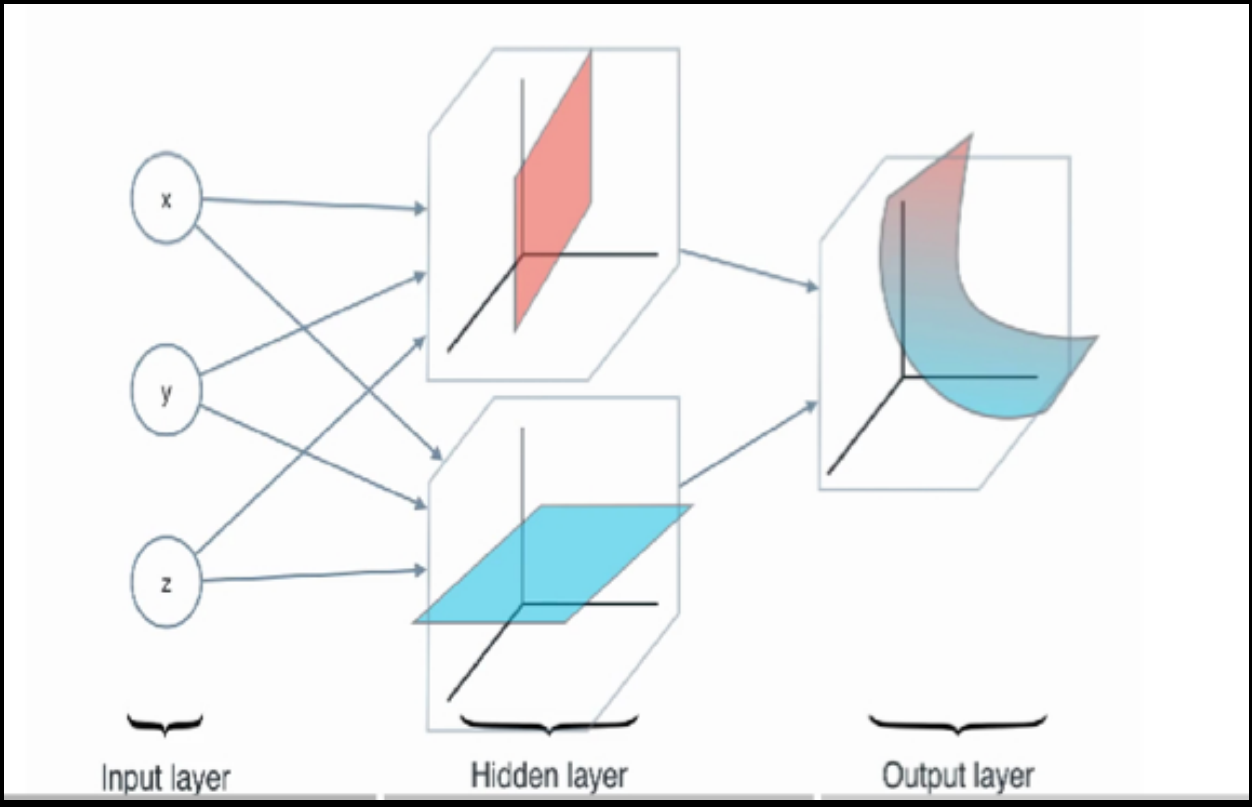


Region Merging

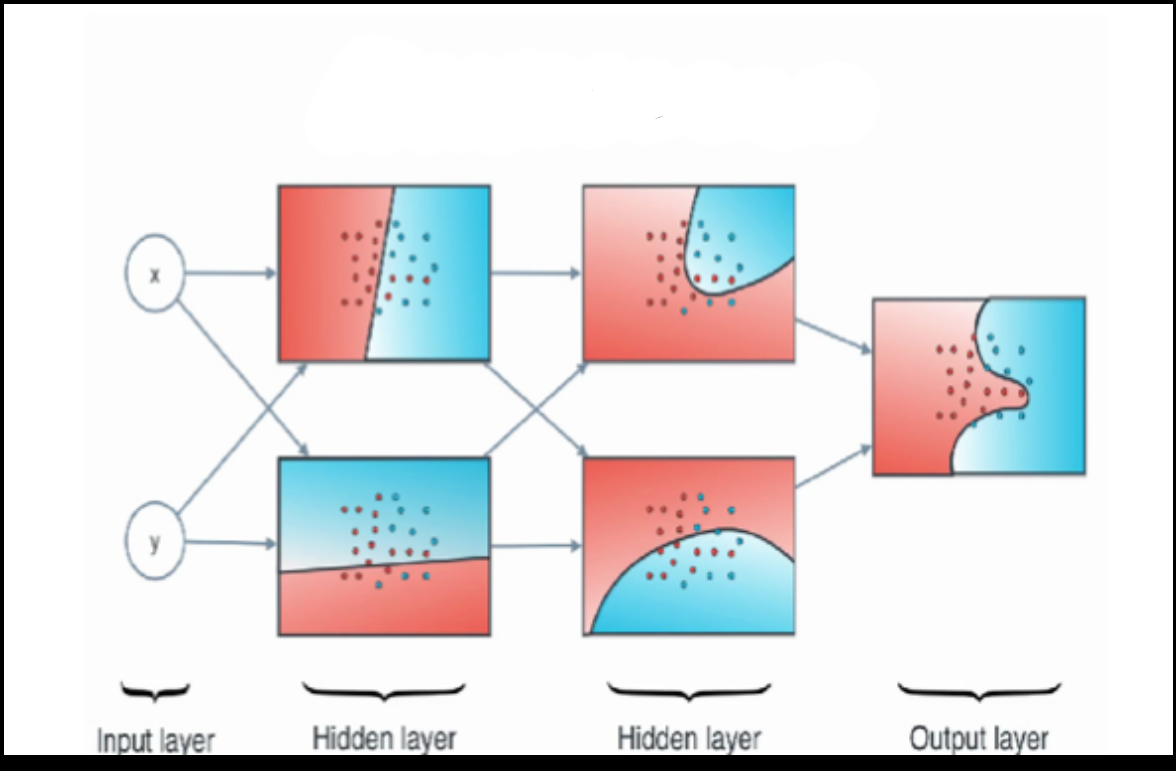


Adding more neuron in a layer

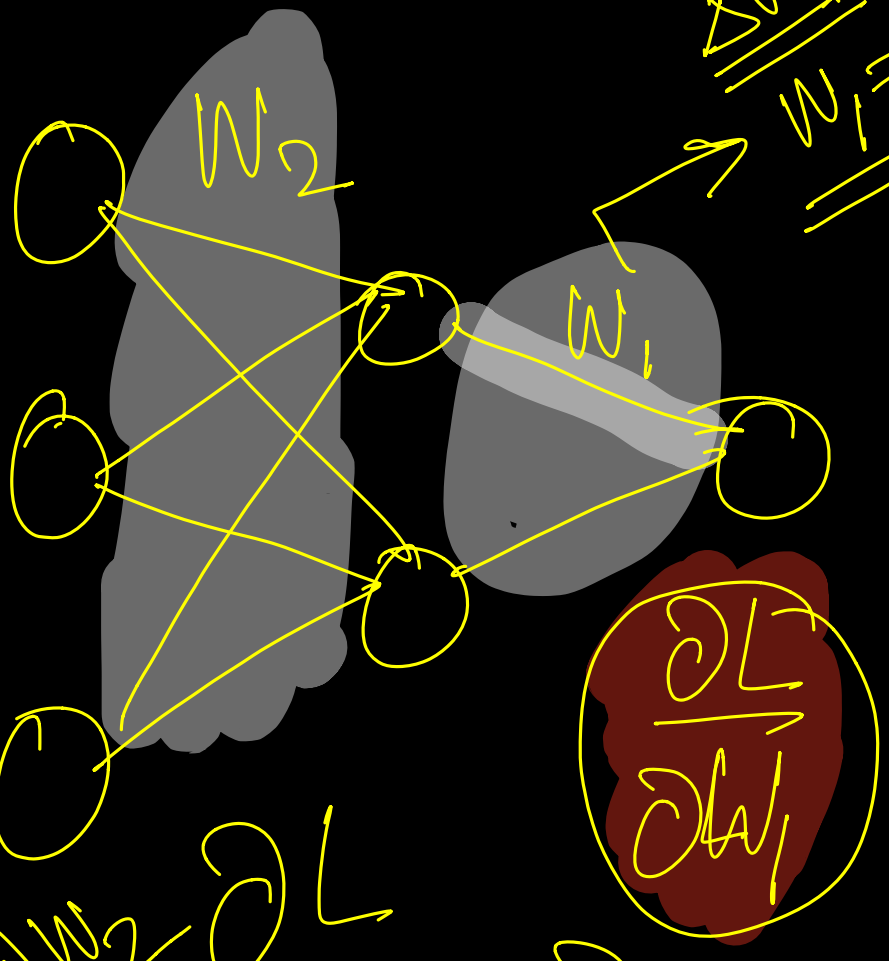
Scaling Vertically



Input dimension > 2



Scaling horizontally.



$$W_2 = W_2 - \eta \Delta W_2 \frac{\partial L}{\partial W_2}$$

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial W_1} * \frac{\partial W_1}{\partial W_2}$$