→ "Artificial Neural Networks" :→

↳ History of Neural Networks :→

It is one of the very very popular Machine learning frameworks & these are so important today, especially since 2012, that there is a whole area called "Deep Learning" which has become extremely popular part of 'ML' today.

↳ Simplest model which we can call as a Neural Network is "Perceptron" that was designed in '1957' by a mathematician called "Rosenblatt".

We will later even see, how a perceptron is very very similar to a "logistic regression" with small changes.

Note :→ "Perceptron" can be imagined as a "Logistic Regression" with a slightly different "loss-function".

There is a little difference, but intuitively they are very related..

This was in 1957, when perceptron was developed. In 1957 just after 2nd world war, when lot of research was being done on translating Russian to English especially in US during the cold war. So, there is lot of research on what Intelligence is, & how to build intelligent Systems.

Some of the greatest minds of that generation like Allen Turing, the father of Modern Computing, & others were asking the very interesting question.

↳ What is Intelligence?

↳ How do we create it artificially in new modern computers?

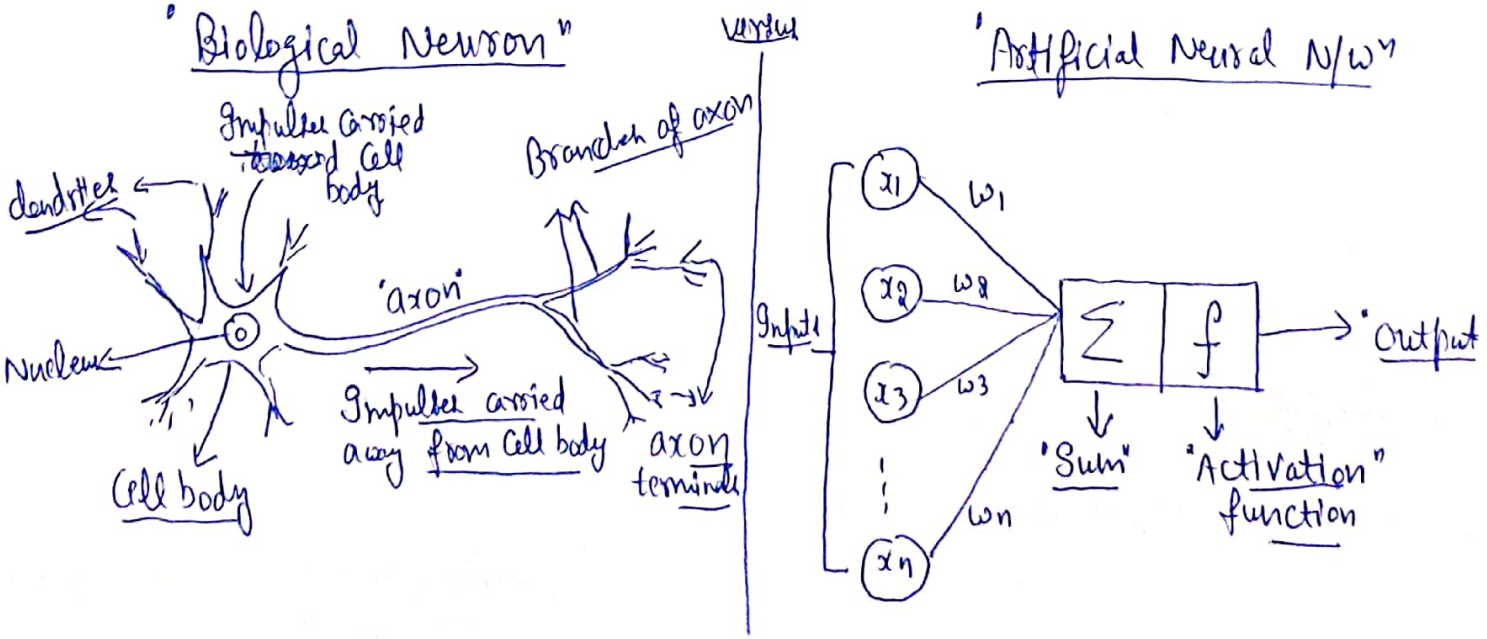
What they did was, they said we have some understanding of "Neuron". (What a neuron is?)

& for all of these people, there is some biological inspiration becoz we know that humans & many animals, like monkeys & even ants have some form of intelligence. There is a lot of inspiration becoz we know thought happens in the brain.

And becoz of lots of research in neuroscience, there is some vague understanding of how the brain works.

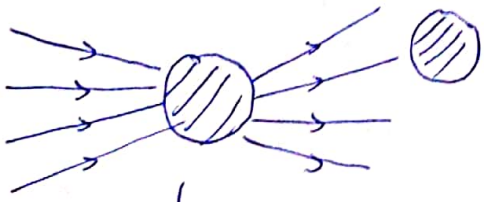
They know that there is something called a neuron which is the basic building block.

Lets see how a neuron operates :→



A neuron (biological neuron) has something called a nucleus & a cell body & there is also something called as dendrites

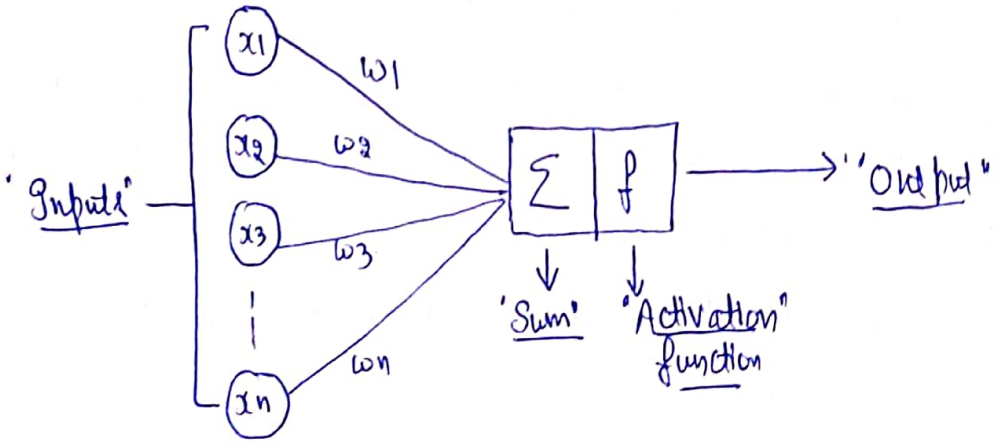
So what happens is when a neuron gets some electrical signals from these dendrites, it does some computation inside of it sends these electrical impulses outside towards some other neurons.



↓ This is the most simple way to understand a biological neuron.

The simplest way to understand it, is, there are some electrical pulses coming, it does some computation inside & then send some electrical signals to other neurons it is connected to. The output generated by a neuron becomes input to other neurons.

This biological structure can be represented mathematically as :→



Imagine if we have inputs like 'x1, x2, x3 ---- xn", people said that can we represent a biological neuron into a mathematical form. Rosenblatt & others, remember are hardcore mathematicians & they were trying to say that can we somehow get a very very simple model of a neuron & put it into a mathematical form (structure)

P.T.O

They said imagine we have some inputs & a cell body. & some of the inputs are more important than others, for that some weights are assigned to these inputs. & this is also understood in the biological neurons. some of these impulses tend to have more importance than others & these are the observations from biological neuron.

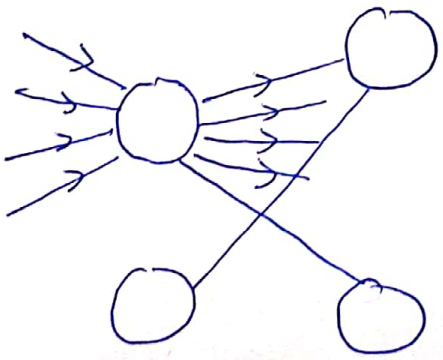
So, we have a bunch of inputs & we want to decide which one to give more importance by using the 'weights'. So, when we have all these inputs, we will sum them up like

$$[w_1x_1 + w_2x_2 + w_3x_3 + --- w_nx_n]$$

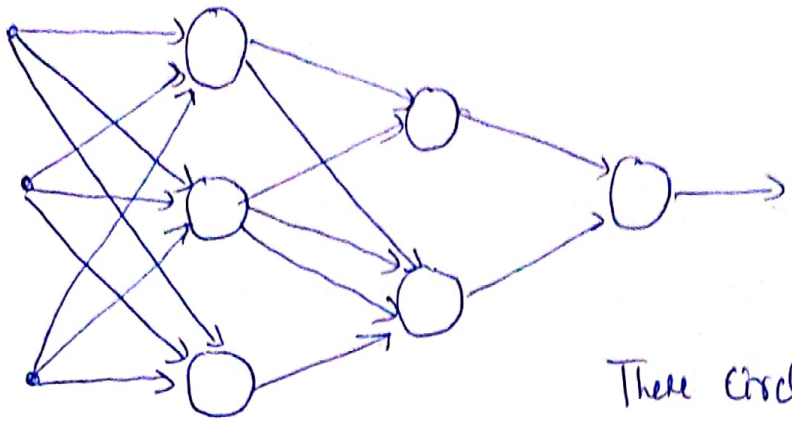
After summating all these inputs, we will apply some function on top of this input summation to get some output & this output can act as an input to other neuron.

This is what, 'Rosenblatt' did & he created a very simplest form of a neuron inspired model called "Perceptron". we will see how a perceptron works, later.

This all happens in 1957. & then came an era where people said, in biology, in actual brains, a neuron doesn't exist on its own. It is connected to other neurons, & there is actually a structure around it. Then they said why we should be limited only by one neuron. Why can't we connect neurons like this



Imagine we are creating a structure of neurons like this



'Network of Neurons'

These circles are neurons & these edges are inp/output.

So, what people said is, we don't have a single neuron, infact what we have is a network of 'interconnected neurons'

Now they said is there a way to actually mimic this 'neural structure'

There were lots of attempts made in 1960's to do this & one such successful attempt was in 1986, where a group of mathematicians, one of whom was Jeff Hinton. who is a legendary figure in modern machine learning.

They came up with an idea called 'back propagation' which is one of the most interesting idea in whole of 'machine learning'.

Note :⇒ 'Back propagation' algo in a nutshell is just ["chain-rule"]. & It is chain-rule of differentiation.

So, they came up with this nice algorithm to train this network of neurons, or how to learn something from this network of neurons.

Around 1980's this whole area of ANN actually emerged, bcoz now we have a n/w of neurons that we can train.

There was a lot of hype around AI in 80's, that people claim. in 10 years from now, we will be having machines that can think like humans

But unfortunately what happened around 1996 was we did not have that computational power that we have today, & we even don't have enough data back then.

So, even some of these algos are very very interesting. Neither did we have computational resources & data to actually make systems which are extremely powerful.

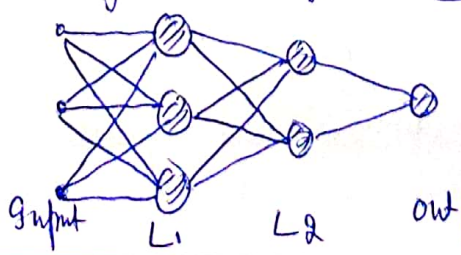
& because of this all of the hype around 'AI' died in late 1990's & that was called AI winter, when funding for AI stopped.

This is when other techniques like SVM's, Ensembles like Random forests, Gradient boosting Decision trees, became very popular after 1995 upto almost 2009. & then techniques evolved around this period & they became super powerful.

Jeff Hinton almost spent 20 years of his life (1986 – 2006) working on how to make neural networks work, better than other algorithms.

Then in 2006 there was this phenomenal paper published by Hinton et al in which he explained how deep neural N/w can be formed & trained.

Back then, if we have to train a network with lots of layer back propagation algorithm fails.,



In b/w input & output we could have many layers & in typical brain we have 100's of layers

So people realized that to really create great algorithms we need very very deep neural networks

[Deep Neural N/w's] By deep we mean the depth.

We have input layer, we have L1 → L2 ——→ L3 → Output layer

. Here we have 3 layers b/w input & output. Imagine if we have 20 layers, how can we train a model like that.

So in 2006. Jeff Hinton came with a phenomenal paper, where he explained how to train deep neural networks, which became foundation of all of deep learning we learn today.

2012, Turning point in DL. (Image net competition)

After 2012, large organization like Google, MS, FB, Amazon even Chinese search engine Baidu, they started noticing about deep learning.

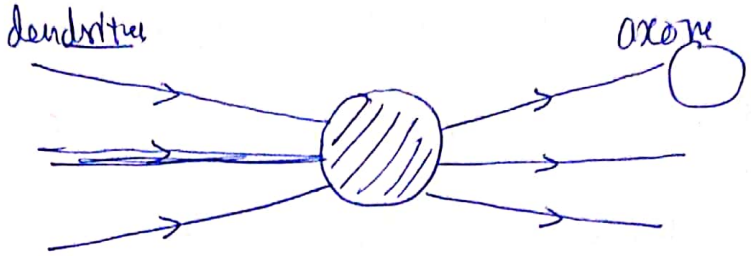
Voice assistant applications like Apple's Siri, Microsoft Cortona, Amazon's Alexa, & google assistant all are powered by advances in Deep learning Algos.

→ How Biological Neurons actually work.

Given an idea of biological neuron, we can now extrapolate that into mathematical models.

We have seen that dendrites, cell body (nucleus) & axons are the most important things in a biological neuron.

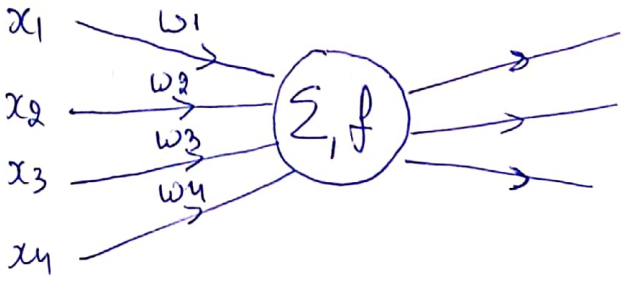
Let's now represent a cell body with a circle, dendrites as edges or lines with incoming directions & axons with outgoing edges.



This is a very very simplified view of a biological neuron. Axons can go ahead & connect with other neurons. But first let's look at it from just one neuron.

So, what happens is, some dendrites are thicker & some are thinner. So basically if a dendrite is thick that implies more weight associated with it.

Mathematically a simple neuron can be represented as:



Since some inputs / dendrites matter more than other inputs. So there are weights associated.

So, larger the weight, the thicker is the dendrite or more informative the input is..

There is a terminology called a neuron is "activated or fired"

So if there is enough input, then a neuron will get fired

This is how a simple neuron works, that if we have enough electrical signals being input then it will get fired or activated hence this function 'f' is often referred to as activation function

& 'w1', 'w2' --- 'wn' are called weights, because if an input edge is having more weight than it is considered to be a thicker dendrite.

Output is represented as:-

$$O = f(w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)$$

$$\boxed{O = f\left(\sum_{i=1}^{n} w_i x_i\right)}$$

Here x_i's are input
w_i's are weights on inputs to the neuron

'Output' — 'activation function' — 'weights' — 'inputs'

Same output is transmitted through each of these Axons or outgoing edges.

So, output of a neuron is basically applying activation function on 'sum of weighted inputs'.

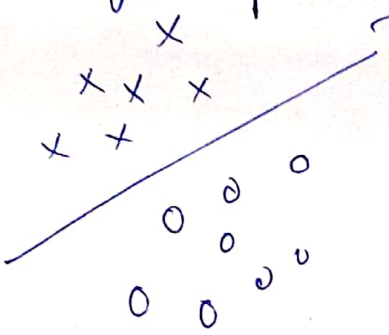
— • —

P.T.O

→ "Logistic Regression & Perceptron"

Let's understand logistic regression from the perspective of a neuron & lets understand the concept of 'Perceptron'

From geometric perspective, we understood that in logistic regression we are creating a hyperplane that separates positive points from negative points.



This is from a geometric or probabilistic perspective.

Now lets try to understand "Logistic regression" from a neural perspective. (from perspective of a neuron)

From mathematical perspective, in 'logistic regression" given any point 'xi", if we want to predict 'yi'. lets called the predicted value as 'ŷi' (predicted value of yi)

In logistic regression we have

$$\hat{y_i} = \text{Sigmoid}\left(w^T x_i + b\right)$$

So, given a bunch of data of {xi's & yi's}

$$D = \{x_i, y_i\}$$ when we train a logistic regression model, we find 'w' & 'b'.

lets assume $x_i \in \mathbb{R}^d$, then w also $\in \mathbb{R}^d$ & b is a real no. i.e. $b \in \mathbb{R}$

In a nutshell the whole relation b/w x_i's & $\hat{y_i}$ is as given below:-

$$\hat{y_i} = \text{sigmoid}(w^T x_i + b)$$

We can write or expand this structure as:-

$$\hat{y_i} = \text{sigmoid}\left(\sum_{j=1}^{d} w_j x_{ij} + b\right)$$

→ This is very very similar to output of a neuron

$$x_i = [x_{i1}, x_{i2}, x_{i3} \dots x_{id}]$$

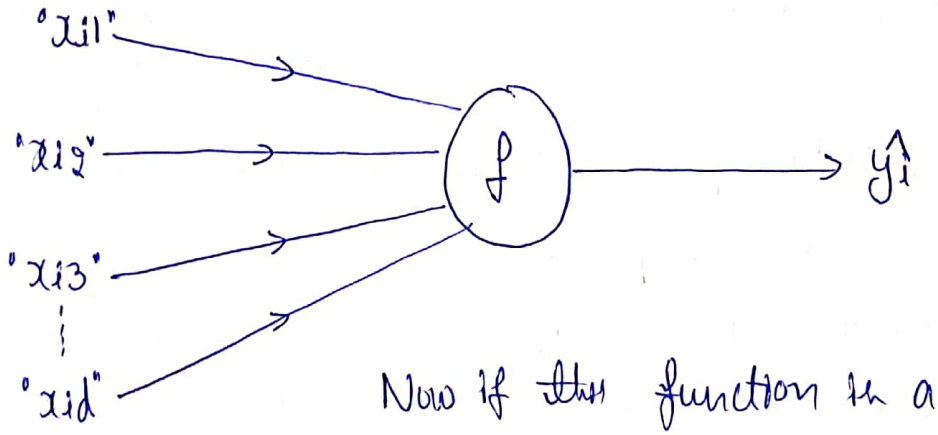
$$w = [w_1, w_2, \dots w_d]$$

Output of a neuron is represented as:-

$$O = f\left(\sum_{j=1}^{d} w_j x_{ij}\right)$$ let's skip the $+b$ part.

↳ This is the functional form of a neuron

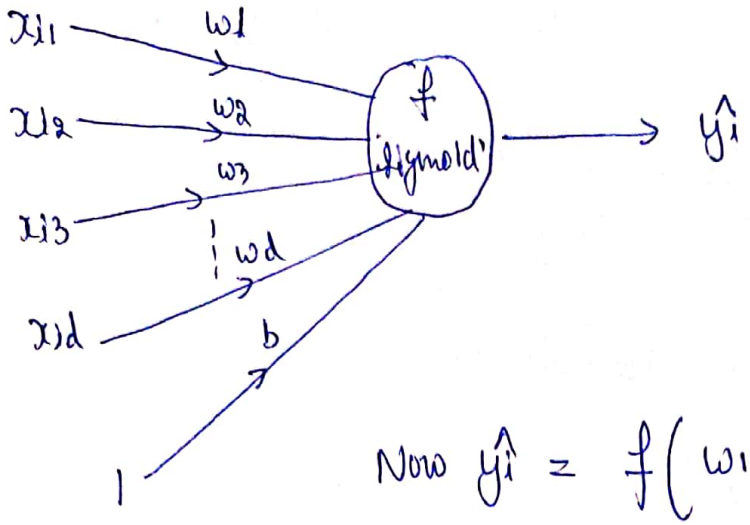
If we have to represent Logistic Regression using neuron, we can do it like



Now if this function is a sigmoid function then we get $\hat{y_i} = \text{sigmoid}(w^T x_i + b)$

"What we do with this 'b'. We - we say apart from all these inputs we have an input '1' whose weight is 'b' as shown.

P.T.O



Now $\hat{y_i} = f(w_1 x_{i1} + w_2 x_{i2} + \cdots w_d x_{id} + 1 \times b)$

$$\hat{y_i} = f(w^T x_i + b)$$

& if 'f' is sigmoid, what we have is exactly the formulation of Logistic regression

This is neuron interpretation of Logistic Regression.

In LR, our task is, when we train 'Logistic-Regression' we find the values of 'w_i' & 'b'.

So in a neural network, we are given a dataset $D = \{x_i, y_i\}$ & our task is to find out the vectors 'w_i' & 'b'

where $i = 1$ to d

Training a neural network is nothing but finding the weights on edges. Becoz if you look at it, these are all inputs. $(x_{i1}, x_{i2} \cdots x_{id})$ & $\hat{y_i}$ is the output. & the task of learning is basically to find these weights $(w_1, w_2, \cdots w_d)$ & b.

So, in a neural network, when we say train a neural network, it simply implies, Compute the weights on edges.

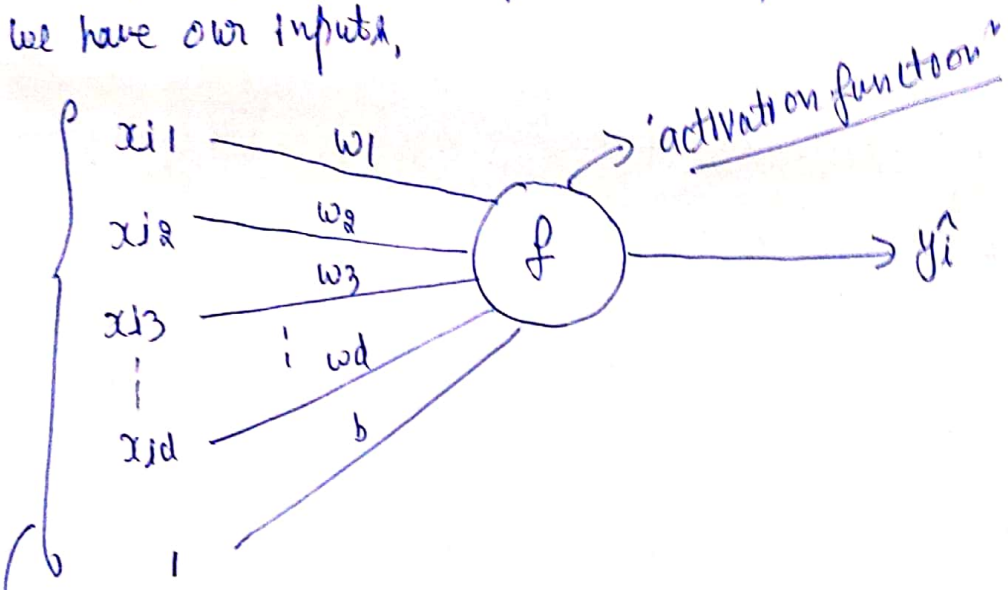
PTO

Now let us understand what is a 'perceptron'.

We have represented a simple logistic regression using a Neuron. The only thing we did was we made activation function as Sigmoid.

Perceptron as an idea was created by "Rosenblatt" in 1957. & idea is very very similar to logistic regression with slight modification.

A simple perceptron looks like this, we have a function we have our inputs,



All these are inputs to the neuron. W's are weights associated with edges.

& $\hat{y_i}$ is output of the neuron

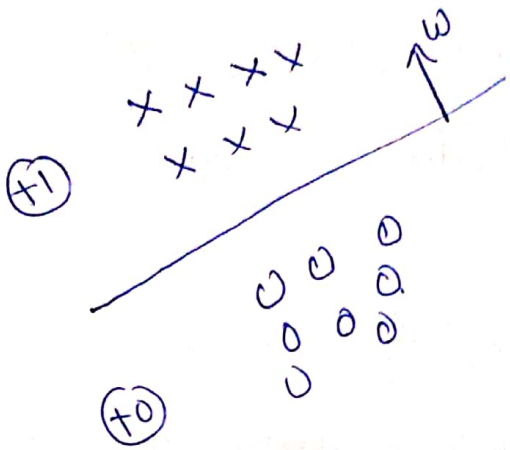
& the only diff. b/w a logistic regression & a perceptron is just the activation function.

Activation function in 'perceptron is :-

$$\begin{cases} f(x) = 1 & \text{if } w^T x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

In LR the activation function is sigmoid. The only thing that changes in Perceptron is the activation function.

Note :- A perceptron is also a 'linear classifier'



[In LR we are using squashing function to do the classification.]

The fundamental difference between "Logistic Regression" and "Perceptron" is the 'loss function'. In logistic regression through sigmoid function we have 'squashing'. In perceptron, we don't have any squashing or sigmoid.

Note :- Perceptron is one of the oldest ML models ever used.

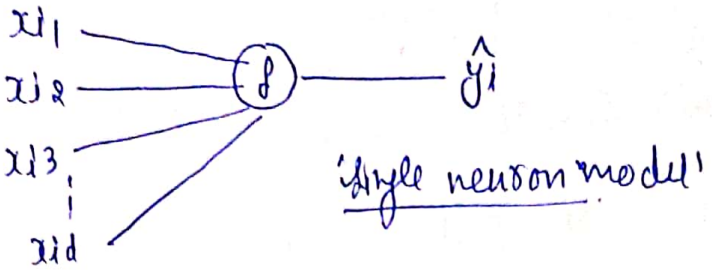
→ MLP (Multilayered Perceptron) :-

We have already seen perceptron, & a perceptron is a very very simplified model of a neuron (style neuron)

& we have also seen that perceptron is very very similar to logistic regression in some way.

Let us now see what is the extension of a perceptron

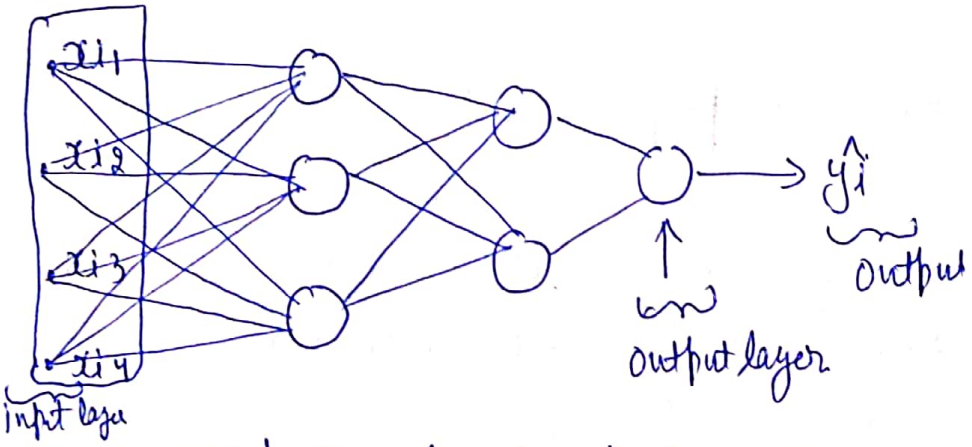
In simple neuron, we have seen, that we have a bunch of inputs and an output.



'Single neuron model'

People said, what if, instead of a single neuron, we have a bunch of connected neurons, that where, we get the name neural network, beoz it is a network of interconnected neurons.

Let's say we have some input points like $x_{i1}, x_{i2}, x_{i3}, x_{i4}$ & people said what if we have multiple neurons instead of a single one



This is called Neural Network, because here we have a collection of neurons connected together

The layers inbetween input & output layer are called hidden layers beoz they are hidden b/w input & output.

In a simple perceptron, we have an input layer & an output layer & nothing in between.

A neural network is often referred to as a multi-layered perceptron, because there are multiple layers here & each of them can be thought of as a perceptron.

Again we have some functions, they could be same or diff..

for simplicity let's call first hidden layer as 'L1' & second as 'L2'

So, this is the structure of a simple neural N/W or a MLP.

Now the big question here is, Why should we care about MLP's?

There are two arguments to this question

1) 'Biological inspiration'

In neuroscience when people started directly brains, they realized that the neural structures within brain ~~is like~~ are actually connection of lots of neurons interconnected in a very very complex ways. There are millions & billions of neurons interconnected in a very very smart way.

We know our basic perceptron is inspired by a simple neuron they said what if we can represent the neural structure.

There is an inspiration that, if a single neuron is powerful, then an interconnected set of neurons must be really really powerful.

Now the big question is how we can implement an MLP.

2) There is also a nice mathematical argument very interesting and simple mathematical argument

Let's take a simple problem of regression

$2 * \sin(x^2) + \text{sqrt}(x * 5)$,

$$F(x) = 2\sin(x^2) + \text{sqrt}(5x)$$

$f_1 \rightarrow$ add( )
$f_2 \rightarrow$ square( )
$f_3 \rightarrow$ sqrt( )
$f_4 \rightarrow$ sin( )
$f_5 \rightarrow$ Mul( )