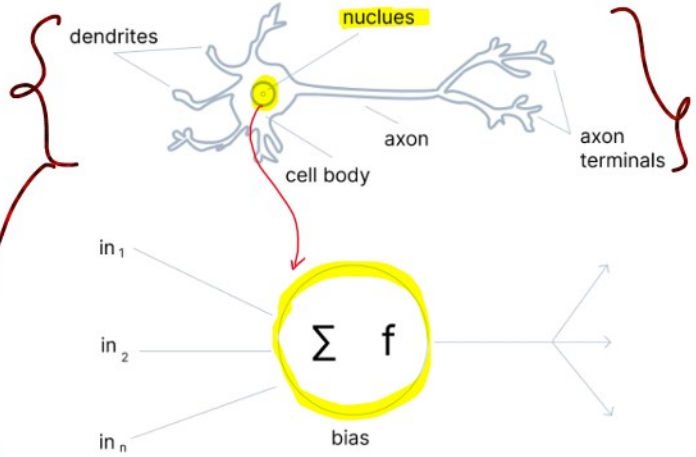
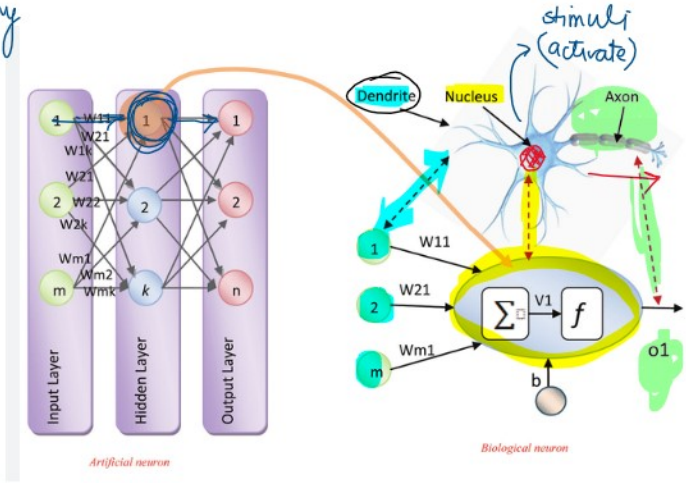
Activation Functions

08 September 2024 20:04

Activation Function helps the neural network to use important information while suppressing irrelevant data points

- Activation function decides whether a neuron should be fired or not (activated)





- Depending on the nature and intensity of these input signals, the brain processes them to decide whether the neuron should be activated or not.

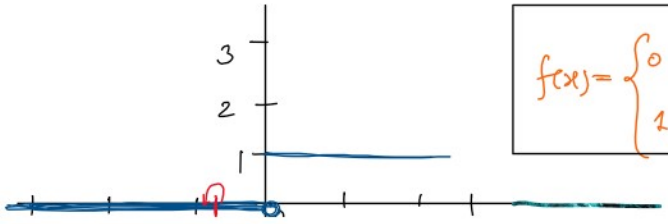
} purpose / role of activation function.

In deep learning, activation function serves this purpose.

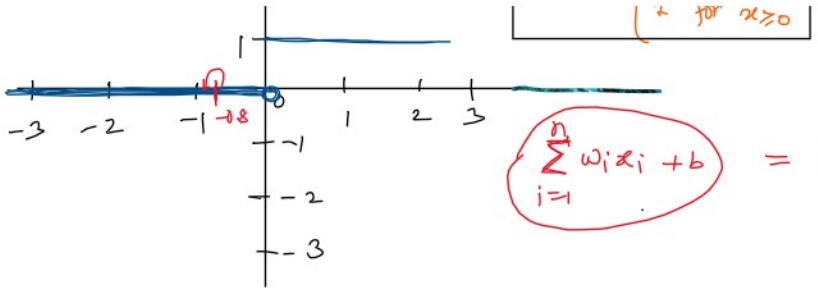
- Activation function is to transform summed weighted input from a node into an output value $\left(\sum_{i=1}^{n} w_i x_i\right)$ to be fed to the next hidden layer or output.

① Binary Step Function

- it depends on a threshold value which decides whether a neuron should be activated or not.



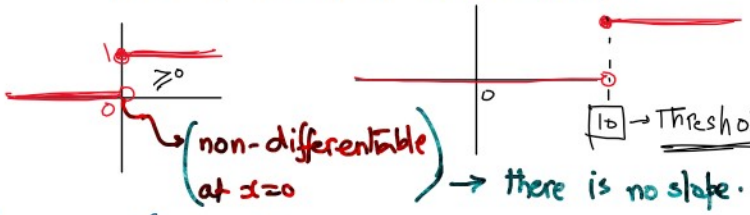
$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$



$\sum_{i=1}^{n} w_i x_i + b = \begin{cases} \geq 0 \\ < 0 \end{cases}$

Protip interview

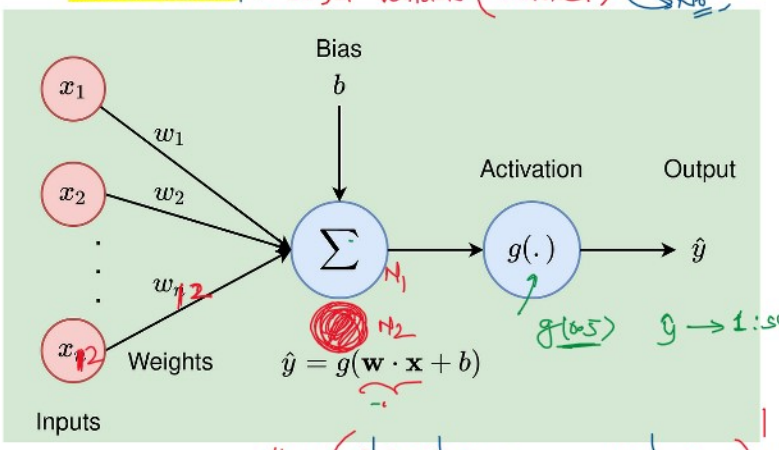
- also known as step function or threshold function



→ there is no slope.

① Binary classification problem

Titanic dataset: Target variable: (survived) → Yes / No



g(0.5) → ŷ → 1 : survived : yes.

N_1: $(w_1 x_1 + w_2 x_2 + \cdots + w_{12} x_{12}) + b \geq 0$ → survived → yes

6.5

N_2: $(w_1^2 x_1 + w_2^2 x_2 + \cdots + w_{12}^2 x_{12}) + b = -0.8$ → 0 Not survived → No.

g(·)

Limitations

① It cannot provide multi-class values – hence it can't be used for multi-class classification problems

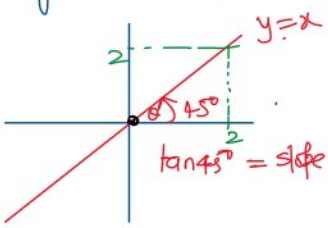
② It doesn't have a derivative at the threshold point which complicates training neural networks using step as the activation function

- gradient-based optimization techniques can't directly optimize networks with this activation [illegible]

- gradient-based optimization techniques can't directly optimize networks with this activation function

② Linear Activation Function

Identity Function $y = x$



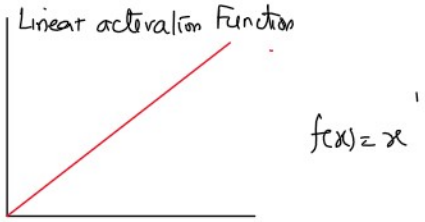
$\tan 45^\circ$ = slope: $m = 1$

$y = mx + c$

$y = 1 \cdot x + 0$

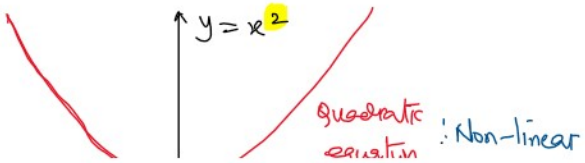
$\boxed{y = x}$

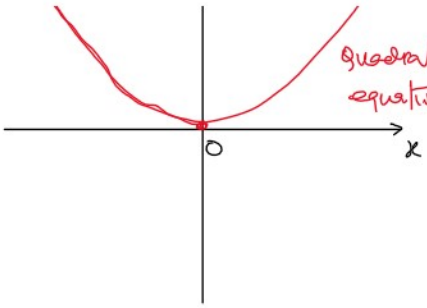
- it is also known as 'no activation' or 'identity function'.
- this activation function doesn't do anything to the weighted sum of the input.



Non-linear activation functions

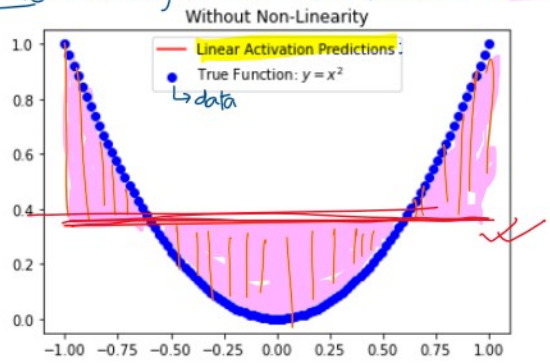
- Linear activation function is simply a linear regression model.
- it does not allow the model to create complex mappings b/w network's input and outputs
- Non-linear activation functions enable the model to learn and capture complex patterns



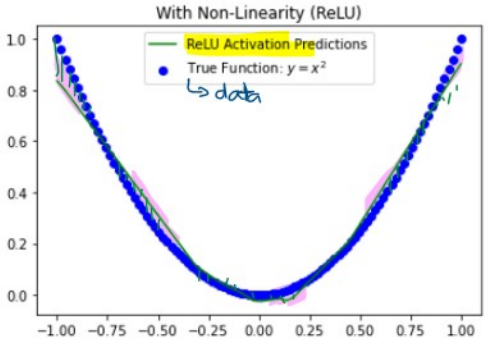


Quadratic equation : Non-linear

scenario #① Modeling a Non-linear function without non-linear activation functions



scenario #② Modeling a non-linear function (data) with non-linear activation function



- Non-linear functions allow back-propagation as the derivative function would be closely related to the input.

- Most of non-linear activation functions allow gradient-based optimization → which means that they are differentiable

  ↓

  compute the gradients of the loss w.r.t each weight and use that information to update the weights

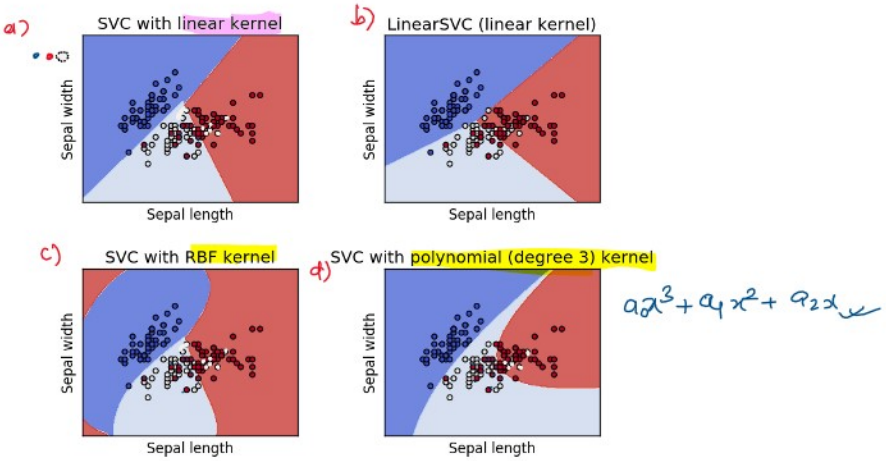
and use that information to update the weights during training.

error = (actual - predicted)

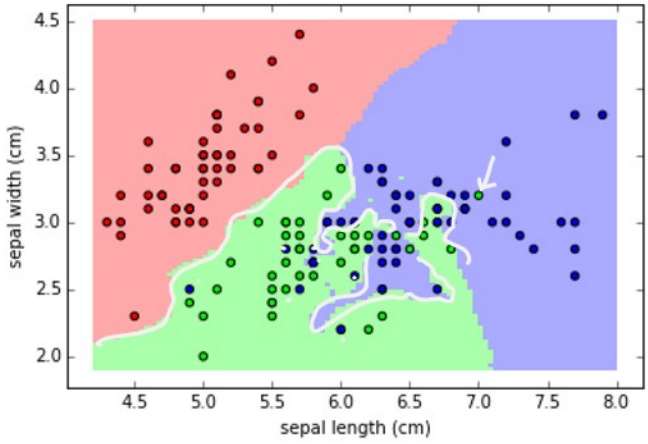
loss function → helps the model to get the best possible fit for data distribution (curve)

Non-linear decision boundaries

Note: sepal width and sepal length Algorithm: Support vector classification



$ax^3 + a_1x^2 + a_2x$



digit recognition

1 2 3 4 5

0-255

Nonlinearity activation function helps in back propagation

$y = 2x + 3$ | $y = x$

$\frac{dy}{dx} = 2$ | $\frac{dy}{dx} = 1$

gradient or derivative is constant

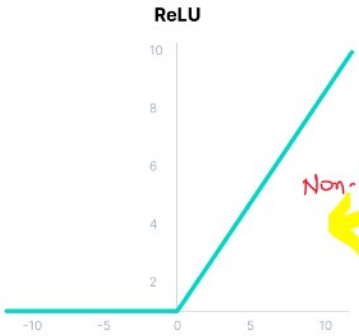
- If all the layers are linear activations, the derivative of the output w.r.t any input or weight will be constant which basically means that the model can't effectively learn relationships

- However if we introduce non-linearities using non-linear activation functions such as ReLu, each layer will have its own gradient (unique) and that eventually helps each layer to learn useful patterns which contribute to minimizing

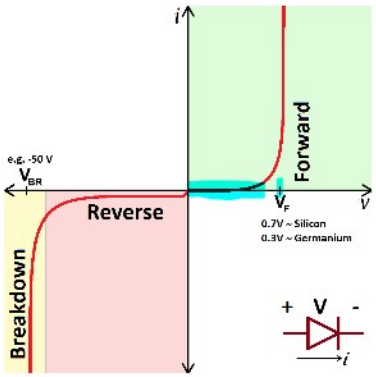
useful patterns which contribute to minimizing the loss function.

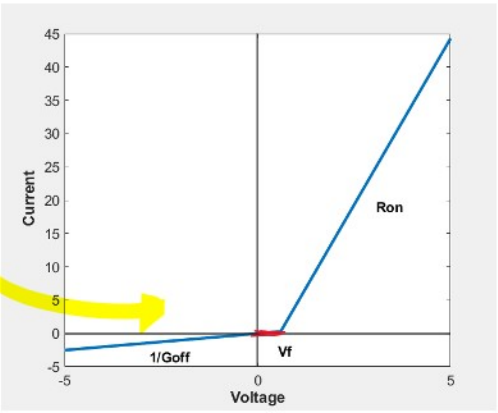
Common Non-linear activation functions

#① ReLU (Rectified Linear Unit)

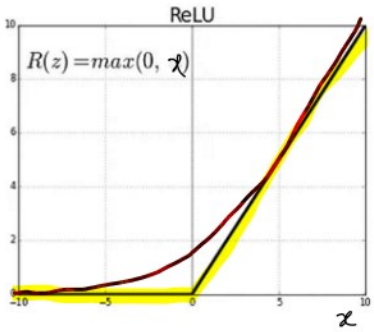


Non-linear





- Although the above graph gives an impression of a linear function however ReLU is a non-linear activation function and has a derivative too.



ReLU is also known piece wise linear function (Non-linear)
It generates the output as the input directly if input is positive, otherwise, it will output zero.

input > 0, output $=$ input

input ≤ 0, output $= 0$

$$f(x) = \max(x, 0)$$

↳ Neurons will be deactivated if

output of transformation is less than or equal to '0'

Ex:

input = -5 output = 0

input = 0 output = 0

input = 5 output = 5

It's simple and computationally efficient, leading to faster training and convergence.

↓

since only a certain no. of neurons are activated, the ReLU function is far more computationally efficient with compared to sigmoid and tanh functions

In general;

$f(x) = \max(x, 0)$ threshold is '0' (without)

↓ ↓ T

$$f(x) = \begin{cases} x & \text{if } x > T \\ 0 & \text{if } x \leq T \end{cases}$$

here $T = 0$

In case, $T = 10$ [with threshold T=10]

$$f(x) = \begin{cases} x & \text{if } x > 10 \\ 0 & \text{if } x \leq 10 \end{cases}$$

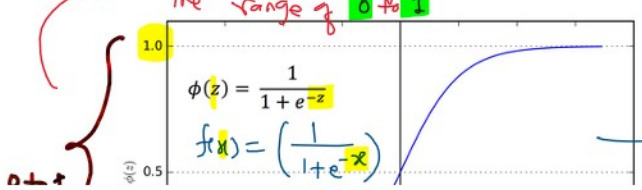
② Sigmoid Activation Function

one of the most common activation functions, used for binary classification functions.

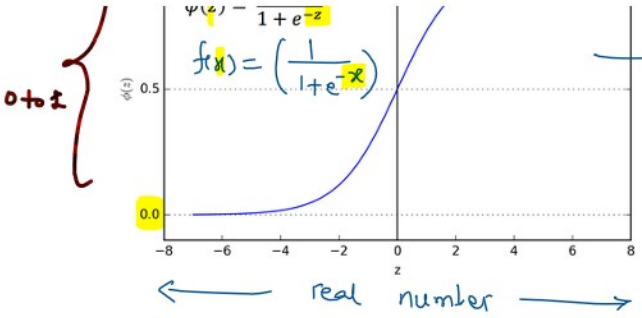
Use-case: Fraud detection — Yes / No

Titonic dataset — survived: yes / survived: No

Probability of anything exists between the range of 0 to 1



→ function is continuous and diff...



function is continuous and differentiable

Note: softmax activation function is a more generalized activation function which is used for multiclass activation function

vanishing gradient problem

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

↓ gradient : w.r.t x

$$f'(x) = \frac{d}{dx}\left[(1+e^{-x})^{-1}\right]$$

$$f'(x) = -1(1+e^{-x})^{-2} \cdot \frac{d}{dx}(1+e^{-x})$$

$$f'(x) = \frac{-1}{(1+e^{-x})^2} * (0 - e^{-x})$$

$$g(x) = f'(x) = \frac{e^{-x}}{(1+e^{-x})^2}$$

gradient

$$f(x) = \left(\frac{1}{1+e^{-x}}\right)$$: sigmoid

$\frac{d}{dx}\left(x^{-1}\right)$ $(n=-1)$

$= -1(x)^{-1-1}$

$= \left(\frac{-1}{x^2}\right)$

$\frac{d}{dx}(x^n) = n x^{n-1} \cdot \frac{d(x)}{dx}$

$= n x^{n-1}$

https://www.khanacademy.org/math/ap-calculus-ab/ab-differentiation-1-new

https://www.khanacademy.org/math/ap-calculus-ab/ab-differentiation-1-new/ab-2-6a/v/derivative-properties-and-polynomial-derivatives

https://www.khanacademy.org/math/ap-calculus-ab/ab-differentiation-2-new/ab-3-1a/v/chain-rule-introduction

$\frac{d}{dx}(e^x) = e^x$

$\frac{d}{dx}e^{-x} = e^{-x} \cdot (-1) = -e^{-x}$

$\frac{d}{dx}(e^{-x}) = -e^{-x}$

$\frac{d}{dx}(\text{constant}) = 0$

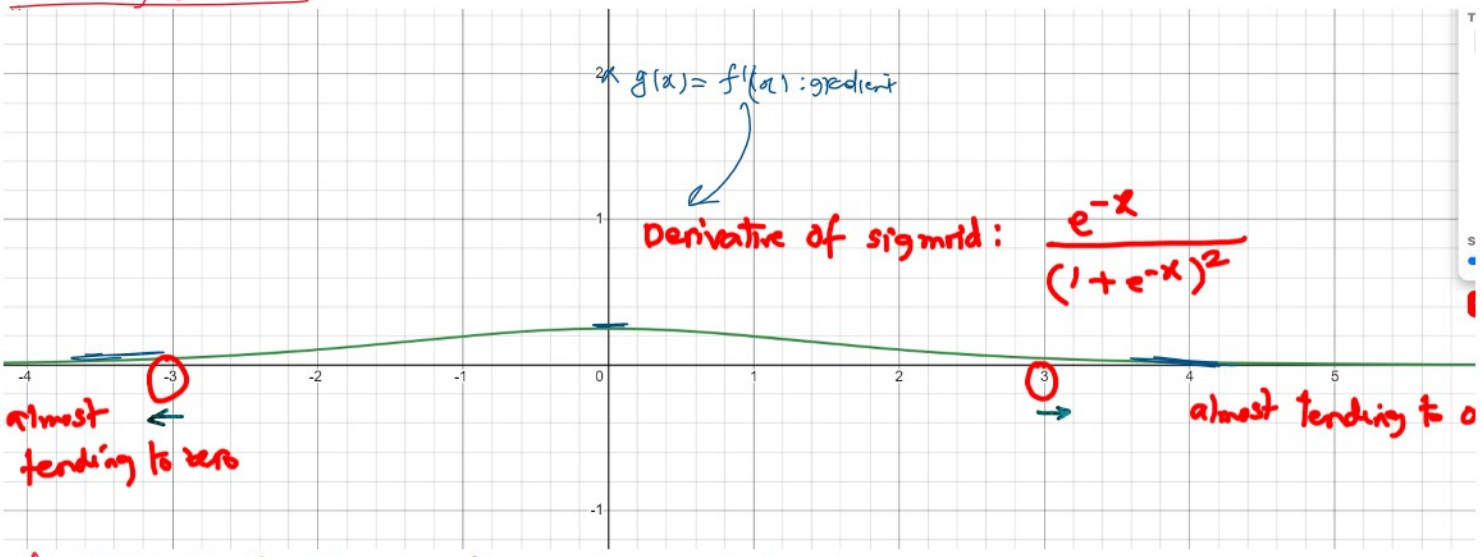
Derivative of sigmoid is:

$g(x) = f'(x) = \text{sigmoid}(x) * [1 - \text{sigmoid}(x)]$

$= \left(\frac{1}{1+e^{-x}}\right) * \left[1 - \frac{1}{1+e^{-x}}\right]$

$$= \left(\frac{1}{1+e^{-x}}\right)\left[\frac{1+e^{-x}-1}{1+e^{-x}}\right] = \frac{e^{-x}}{(1+e^{-x})^2}$$

Vanishing gradient



As we can see from the above figure, the gradient values are only significant for range -3 to 3 and the graph gets much flatter outside this range.
https://www.desmos.com/calculator

↓

As the gradient values approach zero, the network ceases to learn and suffers from the vanishing gradient problem.

For reference

c) Updation rule :

* For $j=0$

$$\theta_0 := \theta_0 - \alpha\left(\frac{\partial}{\partial\theta_0}\left[J(\theta_0,\theta_1)\right]\right)$$

→ is almost zero

Input → by modeler

0.01 * 0.00001 = 0

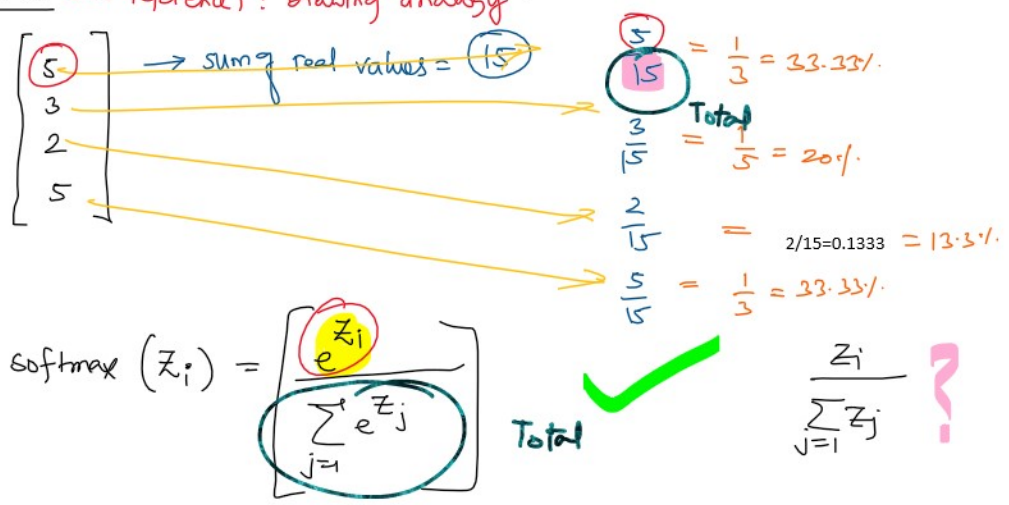
$\theta_0)_{new} = \theta_0)_{old}$

x 100 } time taking

#③ Softmax activation function

- it is a type of activation function that transforms a vector of real-valued numbers into a probability distribution

For ex. (For reference) : Drawing analogy.

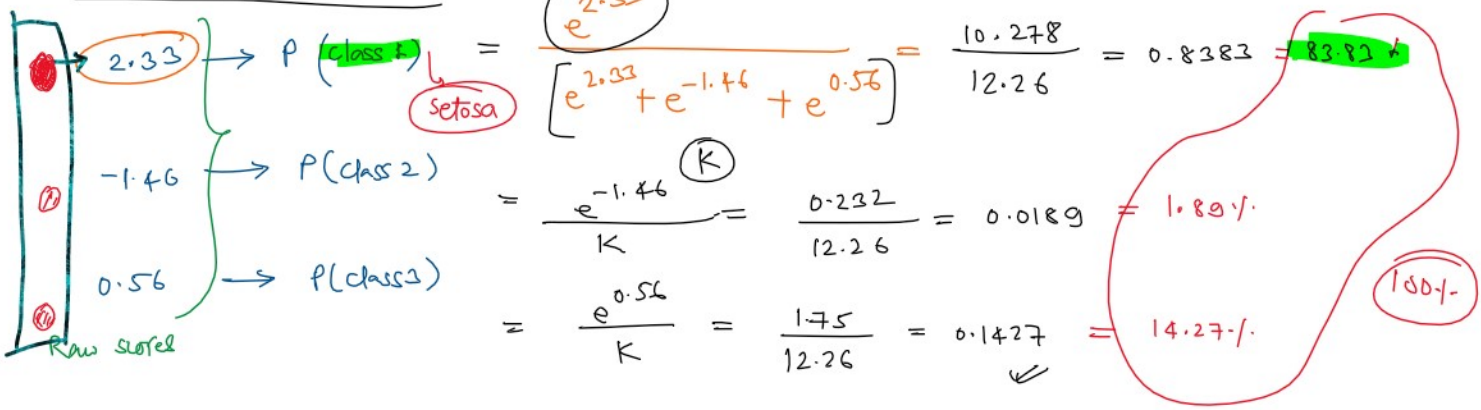


$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{} e^{z_j}}$$ Total ✓

$$\frac{z_i}{\sum_{j=1} z_j}$$?

Example.

Raw Prediction scores:



Key characteristics

1. Normalization: Softmax activation function normalizes the input values into a probability distribution ensuring that the sum of all output values is 1.

2. Exponentiation leveraging exponentiation of the inputs, the softmax function in DL amplifies the difference b/w the inputs, making the largest value more

the ... function ... b/w the inputs, making the largest value more pronounced in the output probabilities

For a vector $z = [z_1, z_2, ..., z_n]$, the softmax function for the i-th element is defined as:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Where:

- z_i is the i-th element of the input vector z,
- e is the exponential function,
- n is the number of elements in the vector.

Application

① Neural Networks : In the output layer of a neural n/w for multi-class classification

For example: In Image classification whether the given image is a cat, dog or car

② Logistic Regression (Multinomial) softmax is an extension of the sigmoid function to perform multi-class classification

Key advantages of softmax function

① Multi-class classification

② Probability distribution – raw scores (logits) to probabilities

③ Differentiable
↳ softmax is smooth, differentiable function, making it suitable for use in back propagation

④ Relative Importance : it emphasizes the relative differences between the Input-values.
↓
one input significantly higher than others → assign a probability

n | · comfy h he l n the > n) ty

w a n ear - | ti the

⑤ stable output