Activation Functions

Activation Functions

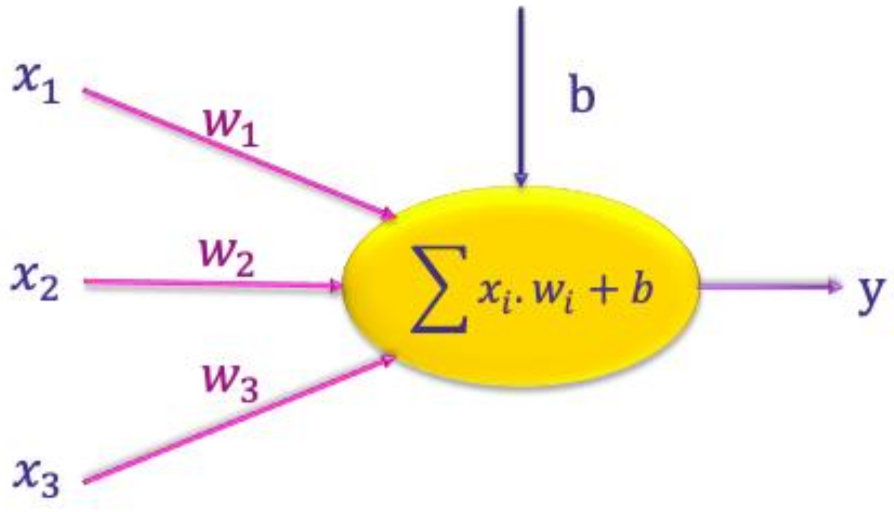
What are Activation Functions ?

- Mathematical functions that determine the output of a neural network
- Connected to each of the neurons and determine if it has to be activated or not based on its value

Activation Functions

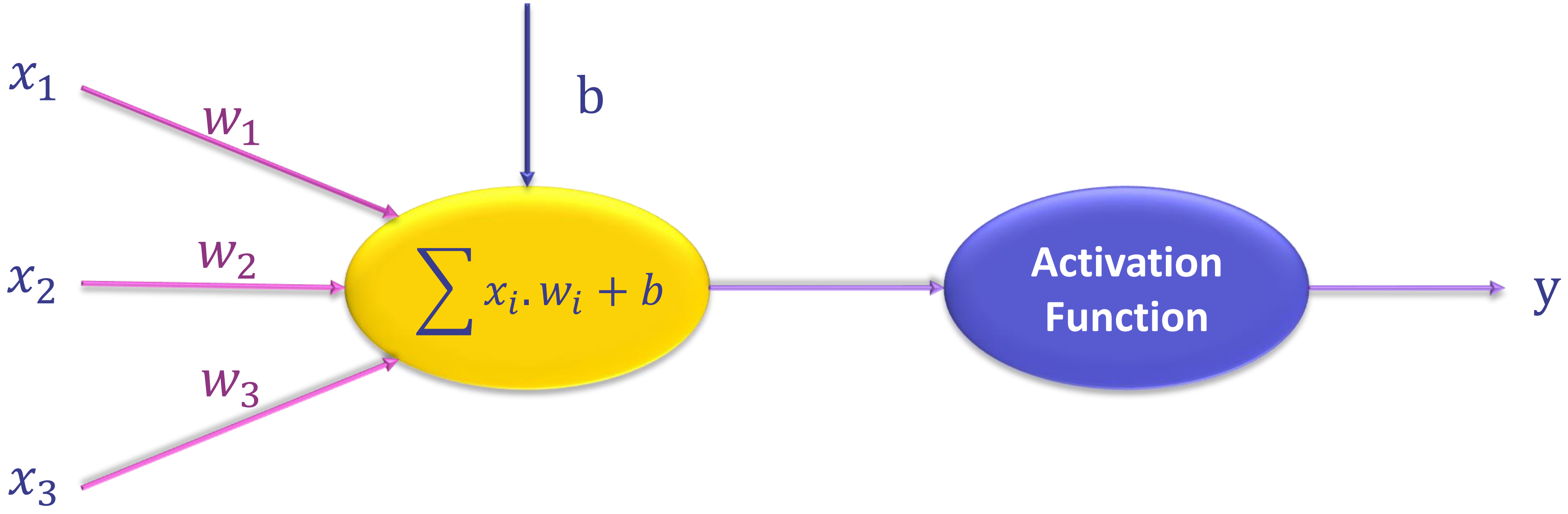
What are Activation Functions ?

- Mathematical functions that determine the output of a neural network
- Connected to each of the neurons and determine if it has to be activated or not based on its value



Activation Functions

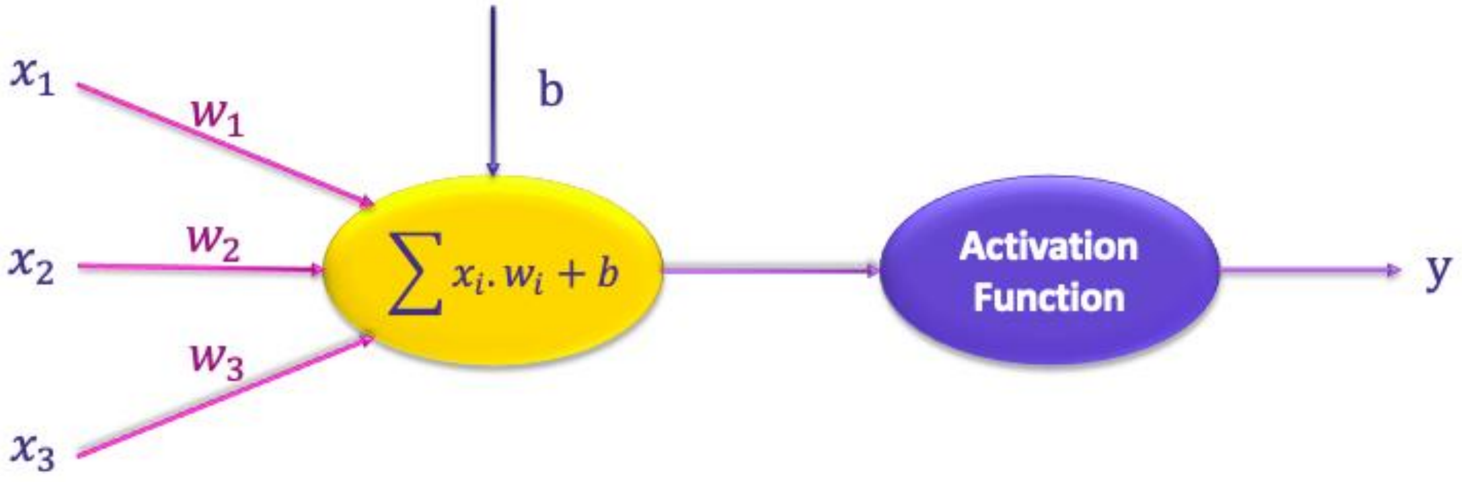
What are Activation Functions ?



Activation Functions

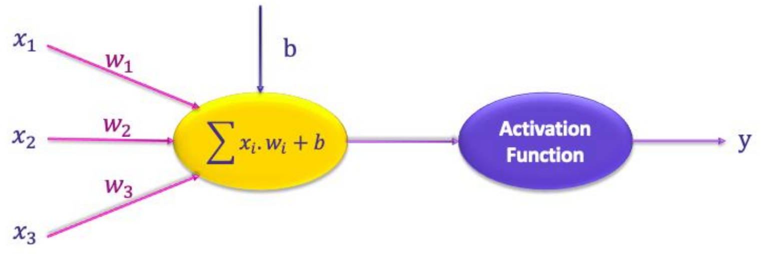
What are Activation Functions ?

The activation function checks the value that it receives, and determines the output of the neuron accordingly



Activation Functions

Why are activation functions useful?

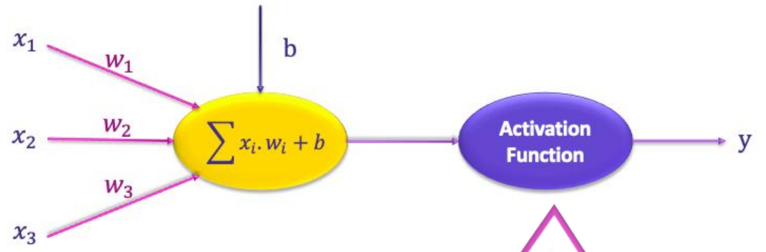


- Add non-linearities to neural networks

→ the neural network can learn to powerfully solve complex problems (ex. image recognition)

Activation Functions

Why are activation functions useful?



- Add non-linearities to neural networks

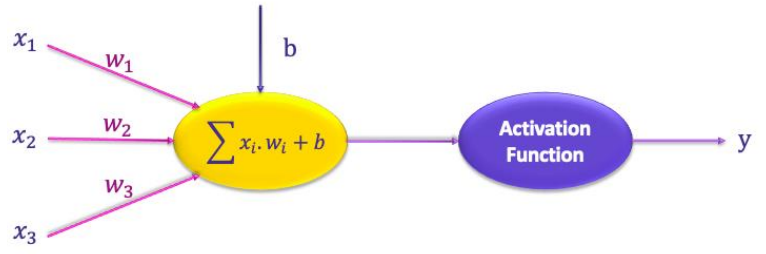
➔ the neural network can learn to powerfully solve complex problems (ex. image recognition)

Without an activation function, the output of a neural network would be a linear relationship between the inputs and their weights.

- ❖ No input management would take place.
- ❖ No non-linearities will be introduced to solve complex problems in analogy with the human brain.

Activation Functions

Why are activation functions useful?

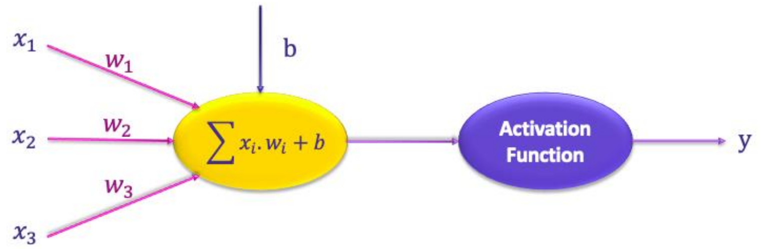


- Help normalize the ouput of every neuron to a range between 0 & 1 or -1 and 1

Activation Functions

Common activation functions :

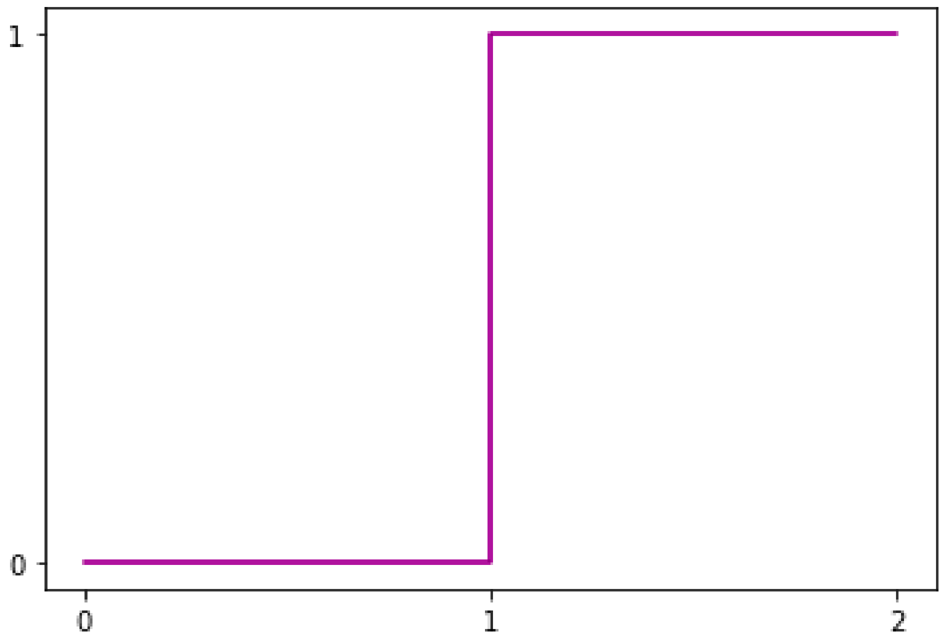
Binary Step Function



- Threshold-based

  ➔ If input ≥ threshold, neuron is activated, and the input value is passed as is

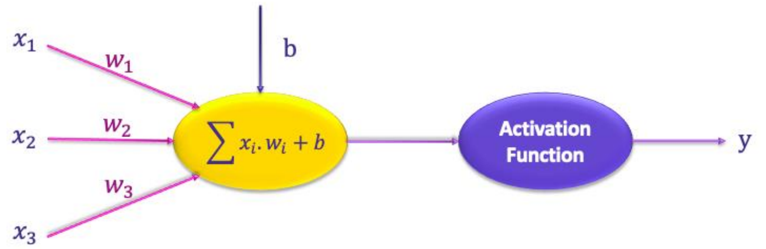
  Otherwise, the neuron is deactivated, and its output is 0

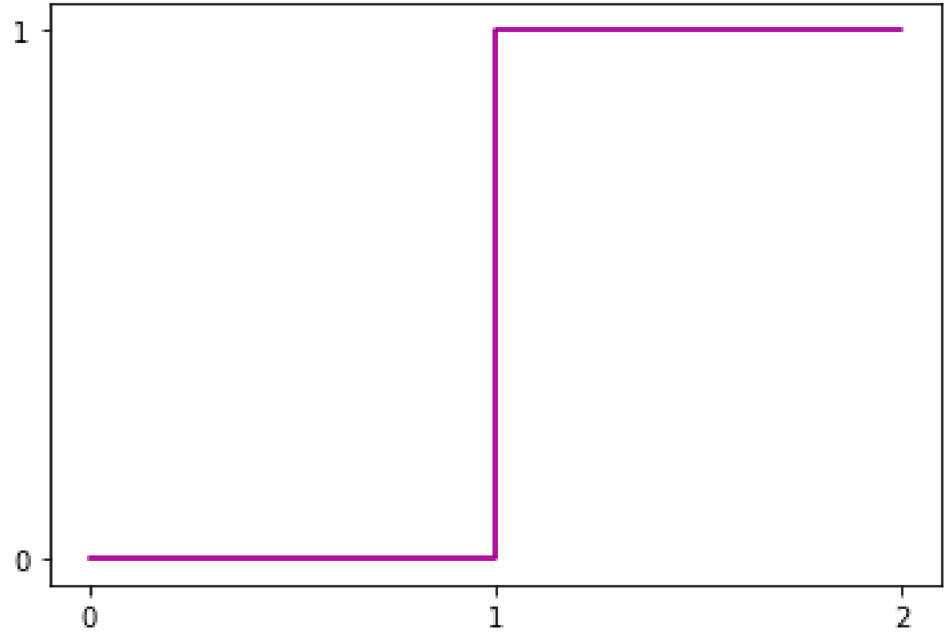


Activation Functions

Common activation functions :

Binary Step Function



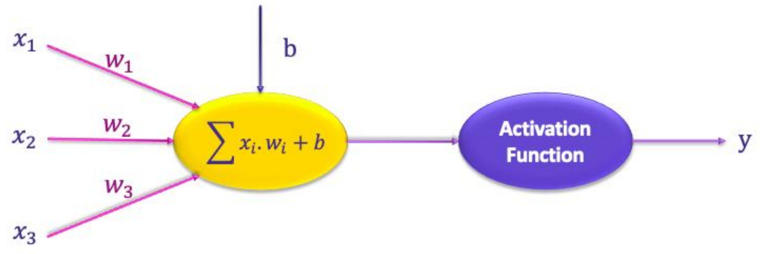


Does not introduce non-linearity

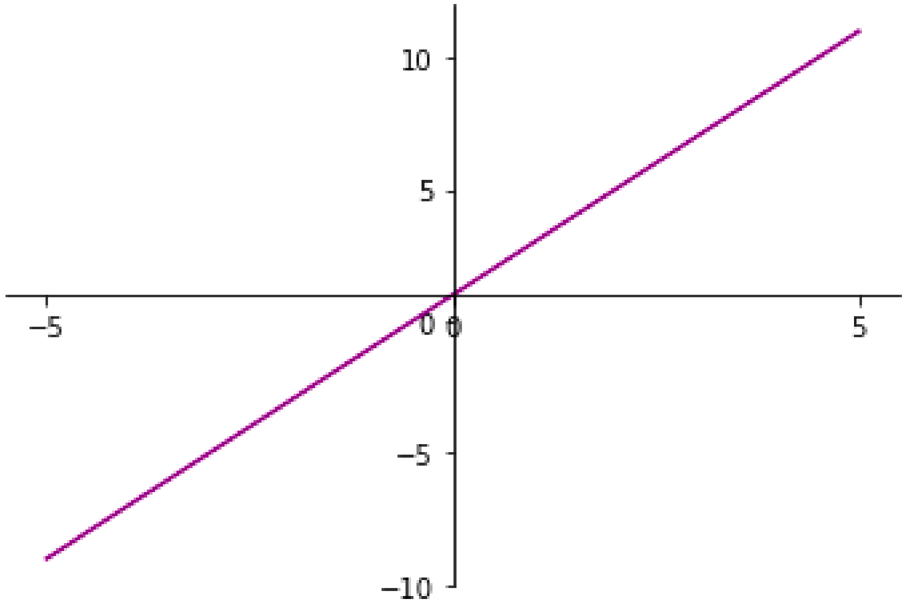
Activation Functions

Common activation functions :

Linear Activation Function



- The output is proportional to the input

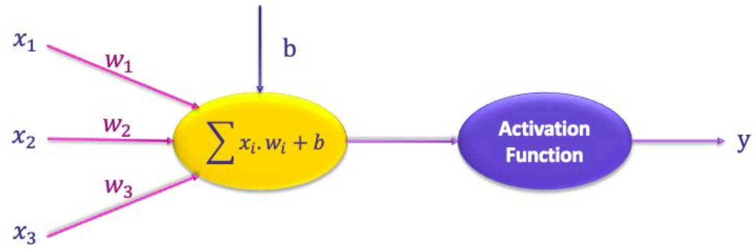


Does not introduce non-linearity

Activation Functions

Common activation functions :

Sigmoid / Logistic Function

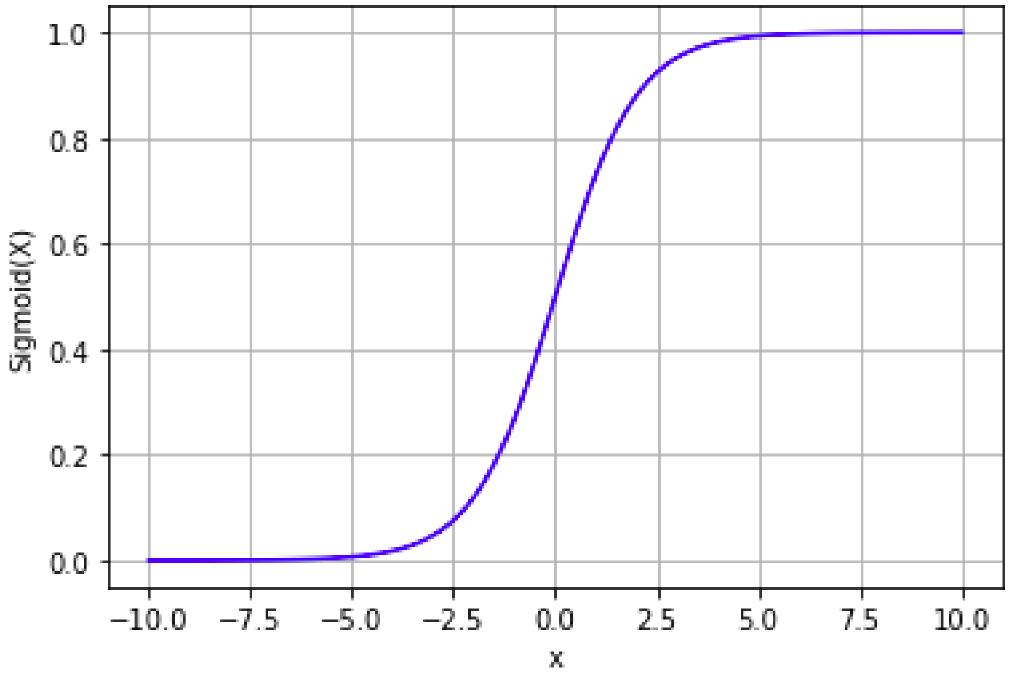


- Output values are between 0 and 1

➔ they can be interpreted as probabilities

- Non-linear

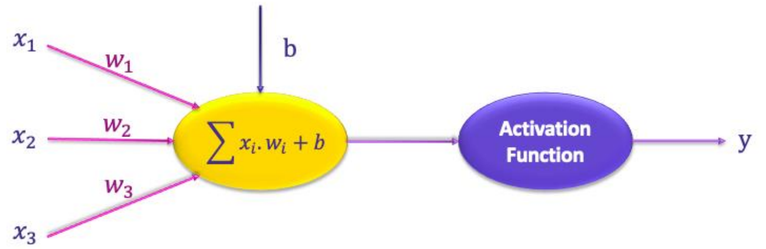
$$S(x) = \frac{1}{1 + e^{-x}}$$

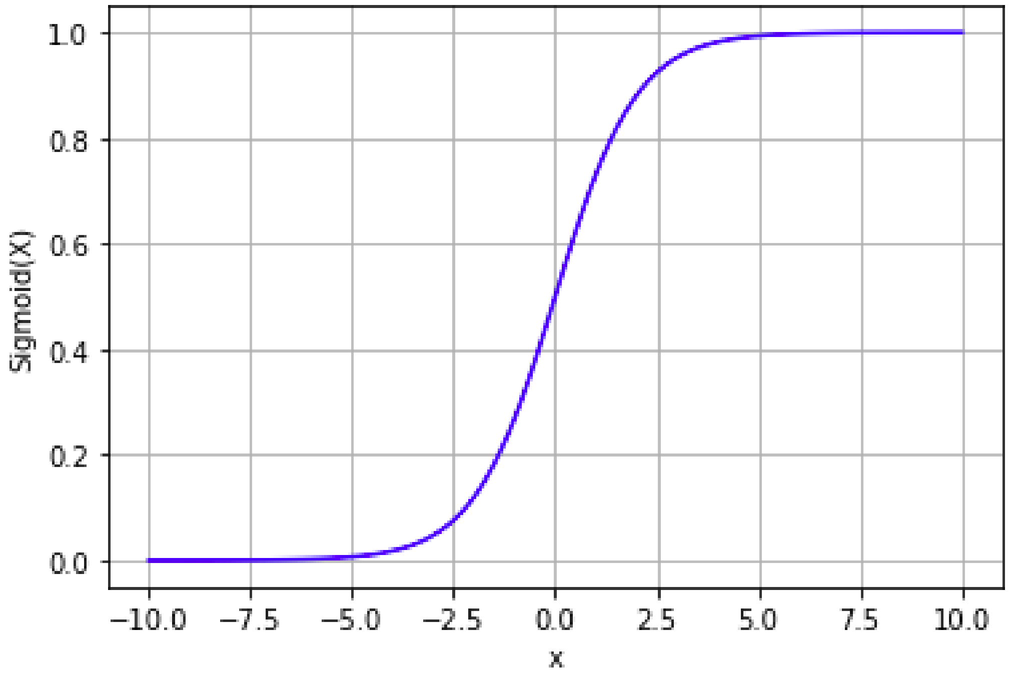


Activation Functions

Common activation functions :

Sigmoid / Logistic Function





- Computationally expensive
- Vanishing gradients (to be seen later)

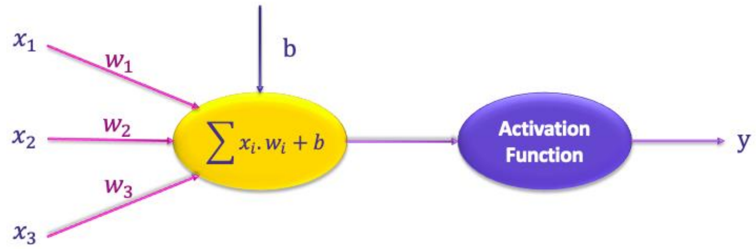
Activation Functions

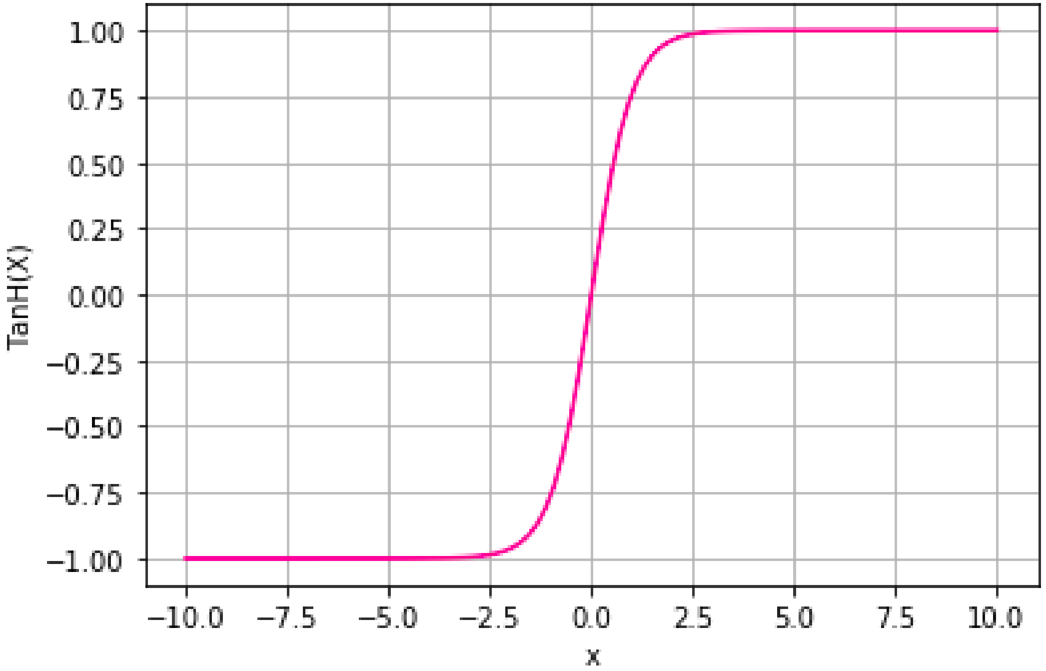
Common activation functions :

TanH / Hyperbolic Tangent Function

- Zero-centered
- Output values are between -1 and 1
- Non-linear

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

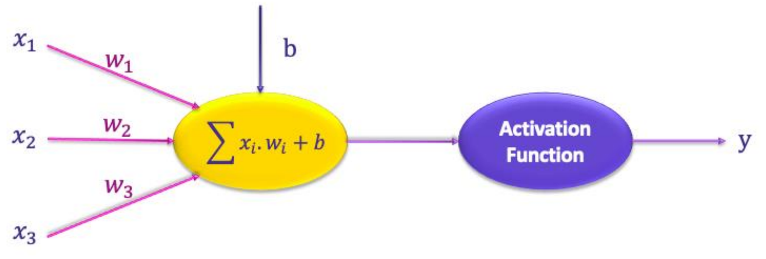


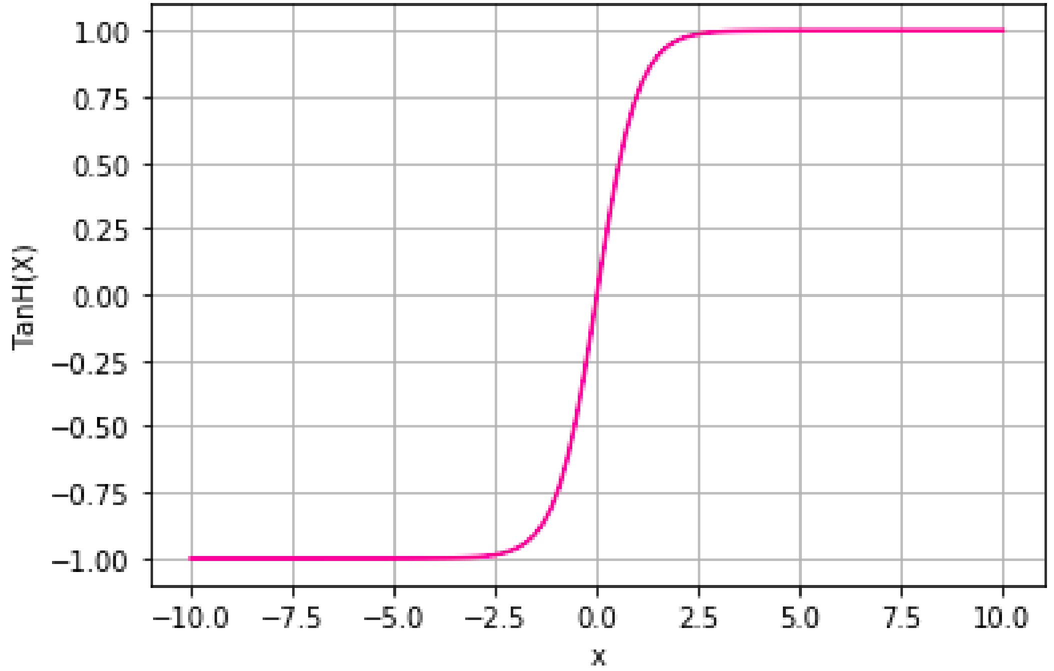


Activation Functions

Common activation functions :

TanH / Hyperbolic Tangent Function



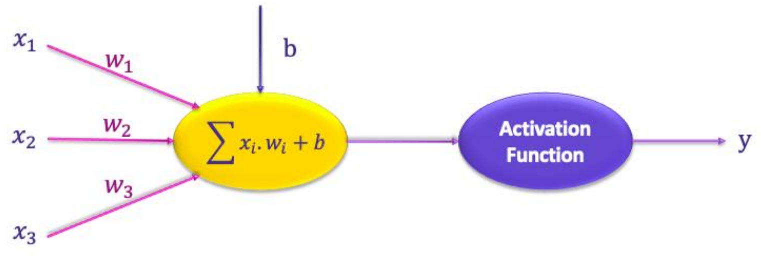


- Computationally expensive
- Vanishing gradients (to be seen later)

Activation Functions

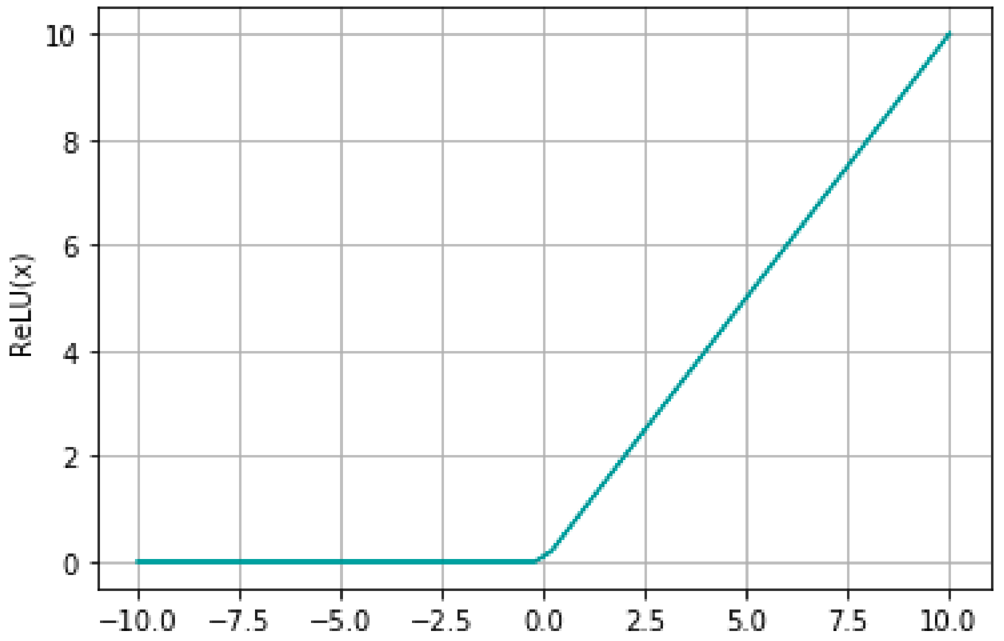
Common activation functions :

Rectified Linear Unit - ReLU



- Returns 0 if input < 0
- Returns the same value as the input if input > 0
- Computationally efficient → allows the network to converge fast
- Non-linear

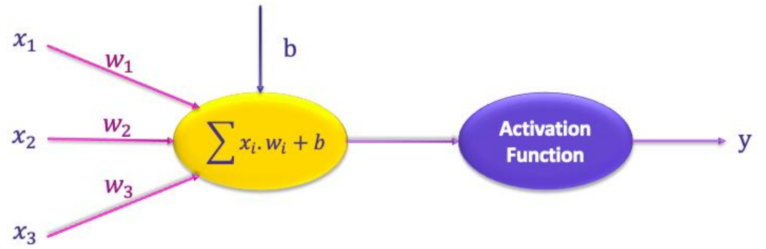
$$ReLU(x) = \max(0, x)$$

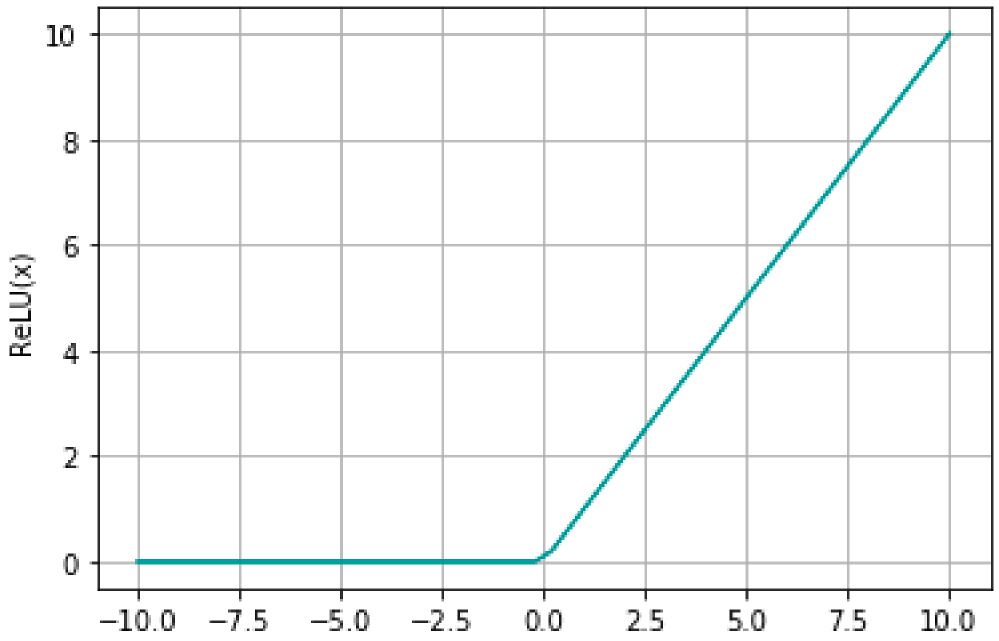


Activation Functions

Common activation functions :

Rectified Linear Unit - ReLU





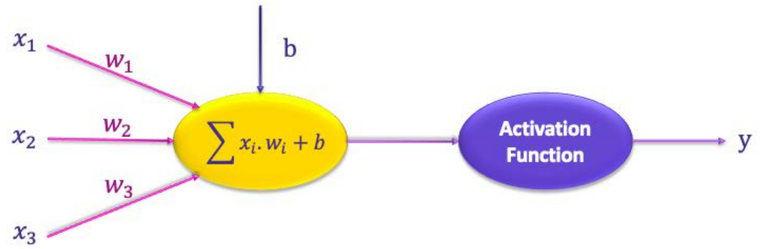
The dying ReLU problem :

When the input is ≤ 0, the network cannot learn

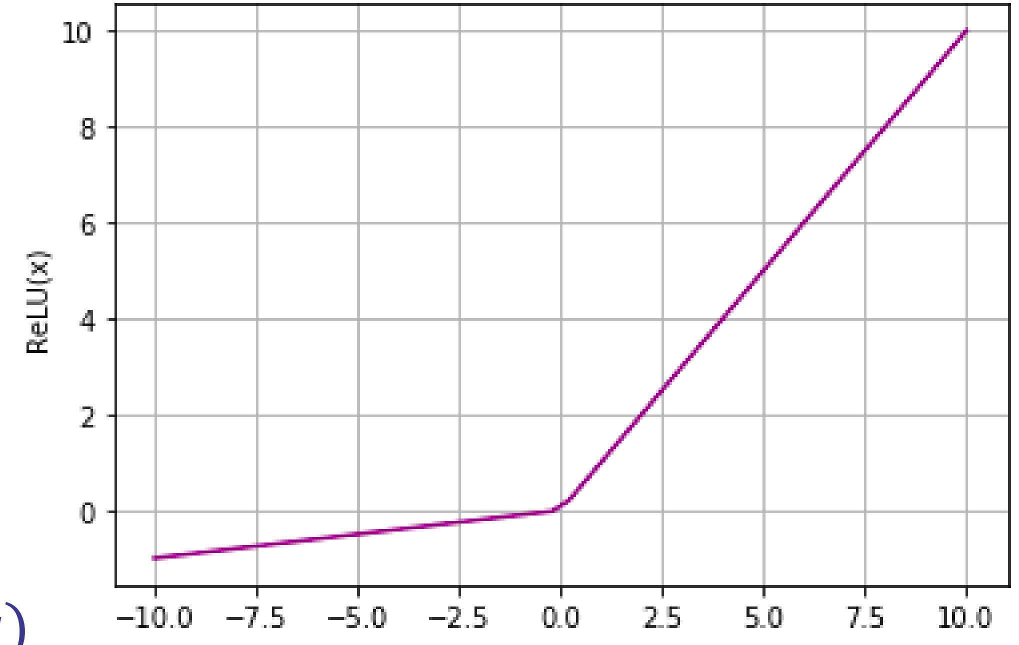
Activation Functions

Common activation functions :

Leaky ReLU



- Prevents the dying ReLU problem → has a small positive slope in the negative side so learning can be done
- Computationally efficient
- Non-linear



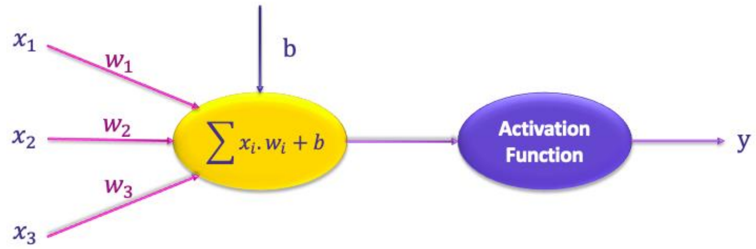
$$Leaky\_ReLU(x) = \max(0.1 * x, x)$$

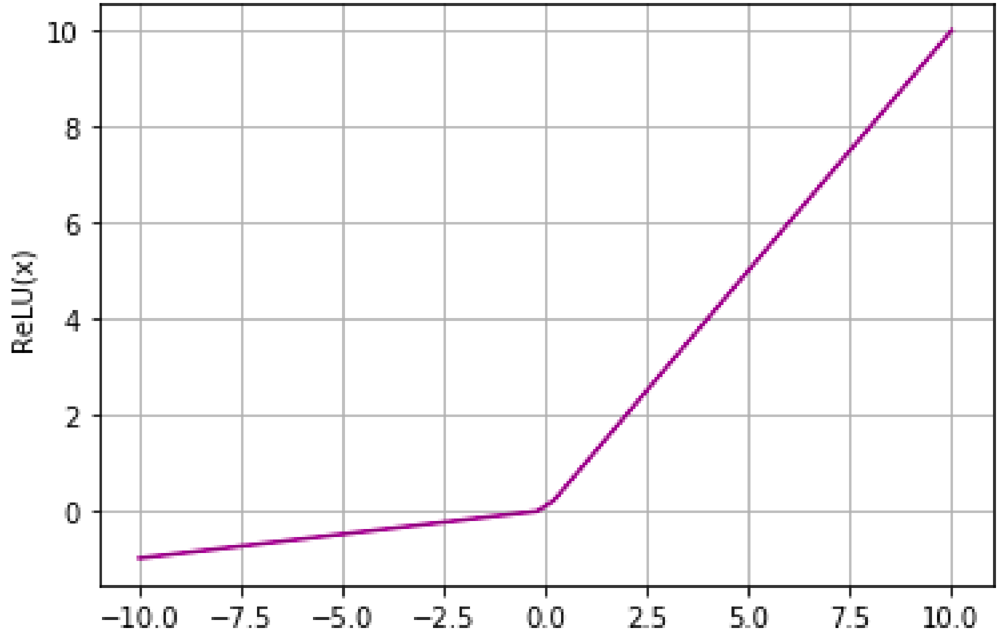
Activation Functions

Common activation functions :

Leaky ReLU

Inconsistent predictions for negative input values

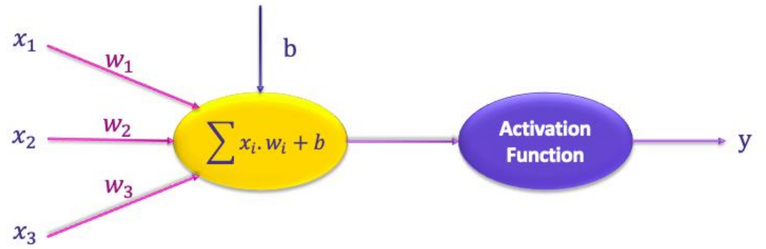




Activation Functions

Common activation functions :

Parametric ReLU (PReLU)



- Its leakage coefficient is a learned parameter

  ➔ its slope is learnable

- Prevents the dying ReLU problem
- Computationally efficient
- Non-linear

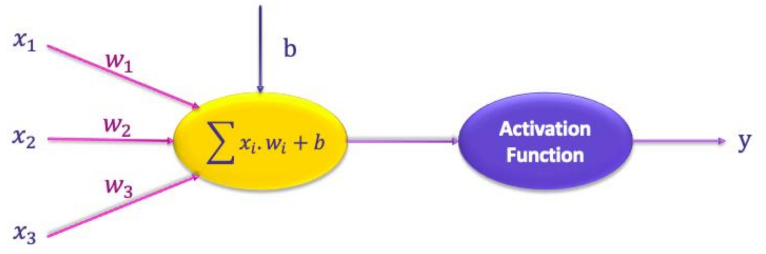
$$PReLU(x) = \begin{cases} x & if\ x > 0 \\ ax & otherwise \end{cases}$$

leakage coefficient

Activation Functions

Common activation functions :

Parametric ReLU (PReLU)

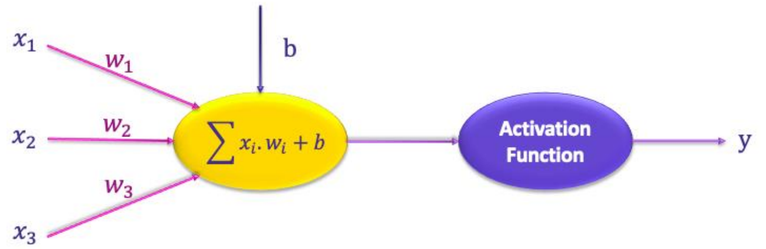


May perform differently in different problems

$$f(x) = \begin{cases} x & if\ x > 0 \\ ax & otherwise \end{cases}$$

Activation Functions

Common activation functions :

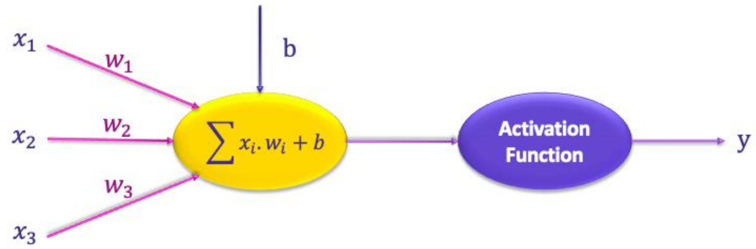


Softmax

- Able to handle multiple classes (normalizes the output of each class between 0 & 1 → classification probabilities)
- The output probabilities sum to 1
- Non-linear
- Typically used for the output layer

$$Softmax(x) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

Activation Functions



Common activation functions :

Softmax

ANN Outputs

$$\begin{bmatrix} 2.1 \\ 3.2 \\ 1.5 \end{bmatrix}$$

$$Softmax(x) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

$$\frac{8.17}{8.17 + 24.53 + 4.48}$$

$$\frac{24.53}{8.17 + 24.53 + 4.48}$$

$$\frac{4.48}{8.17 + 24.53 + 4.48}$$

Probabilities

$$\begin{bmatrix} 0.22 \\ 0.66 \\ 0.12 \end{bmatrix}$$

Activation Functions

Common activation functions :

Softmax

Why does Softmax use exponential functions for normalisation ?

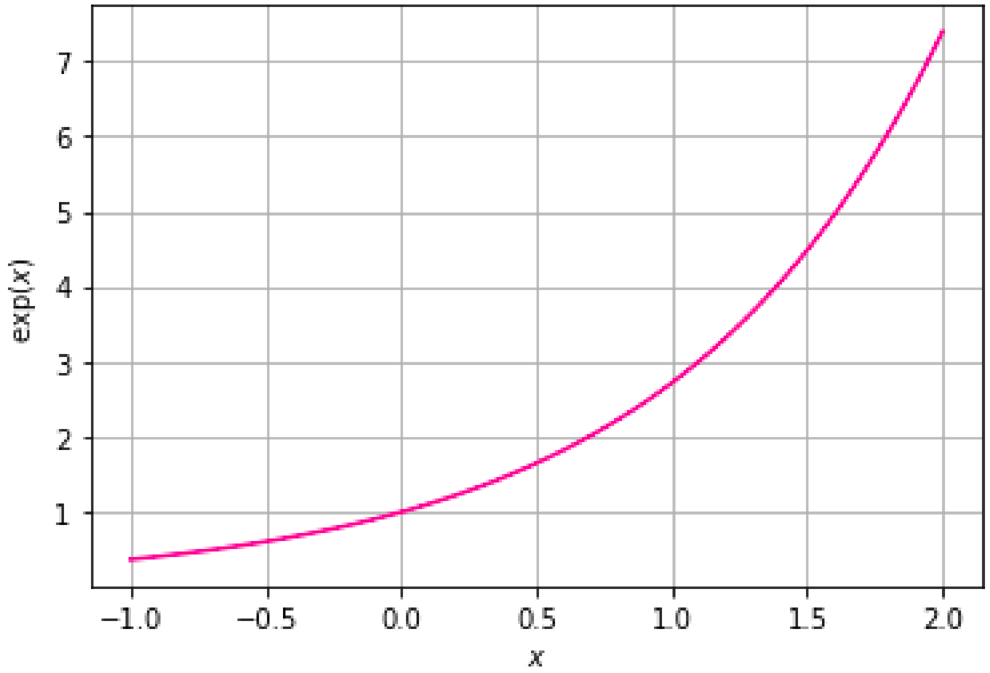
The exponential function makes high values even higher

➔ It makes sure that the probability of the most probable class stands out

$$e^0 = 1$$

$$e^2 = 7.4$$

$$e^4 = 54.6$$



Activation Functions

Common activation functions :

Softmax

Normalisation without Softmax

ANN Outputs

$$\begin{bmatrix}2.1\\3.2\\1.5\end{bmatrix} \rightarrow \begin{matrix}\dfrac{2.1}{2.1+3.2+1.5}\\ \dfrac{3.2}{2.1+3.2+1.5}\\ \dfrac{1.5}{2.1+3.2+1.5}\end{matrix} \rightarrow \begin{bmatrix}0.31\\0.47\\0.22\end{bmatrix}$$

Probabilities

Normalisation with Softmax

$$Softmax(x) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

ANN Outputs

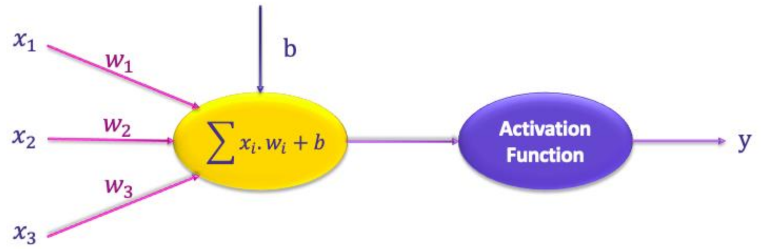
$$\begin{bmatrix}2.1\\3.2\\1.5\end{bmatrix} \rightarrow \begin{matrix}\dfrac{8.17}{8.17+24.53+4.48}\\ \dfrac{24.53}{8.17+24.53+4.48}\\ \dfrac{4.48}{8.17+24.53+4.48}\end{matrix} \rightarrow \begin{bmatrix}0.22\\0.66\\0.12\end{bmatrix}$$

Probabilities

Activation Functions

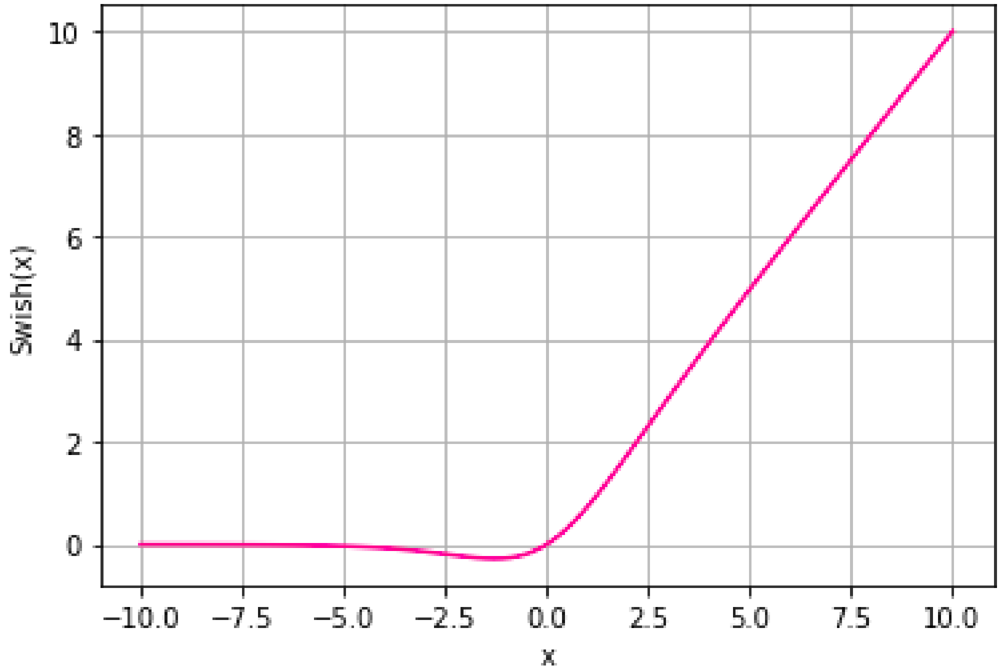
Common activation functions :

Swish Activation Function



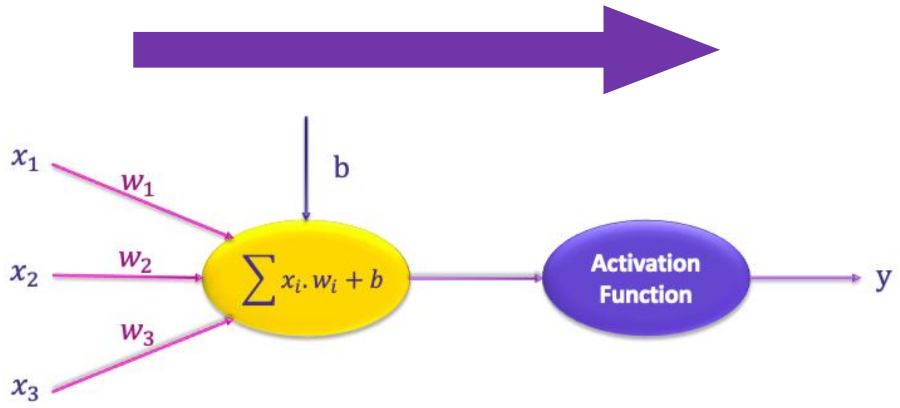
- Discovered by researchers at Google Brain
- Outperforms ReLU for deep networks
- Computationally efficient
- Non-linear

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



Forward Propagation

Forward Propagation



What are Feedforward Neural Networks?

- ANNs where the input data flows in the forward direction through the network
- The data does not flow backwards during output generation
- No cycles or loops exist in this network
- Feedforward Neural Networks support forward propagation

Forward Propagation

What are Feedforward Neural Networks?

At each neuron, two steps take place :

1. Preactivation : The weighted inputs are summed up

2. Activation : The weighted sum of inputs is passed to the activation function

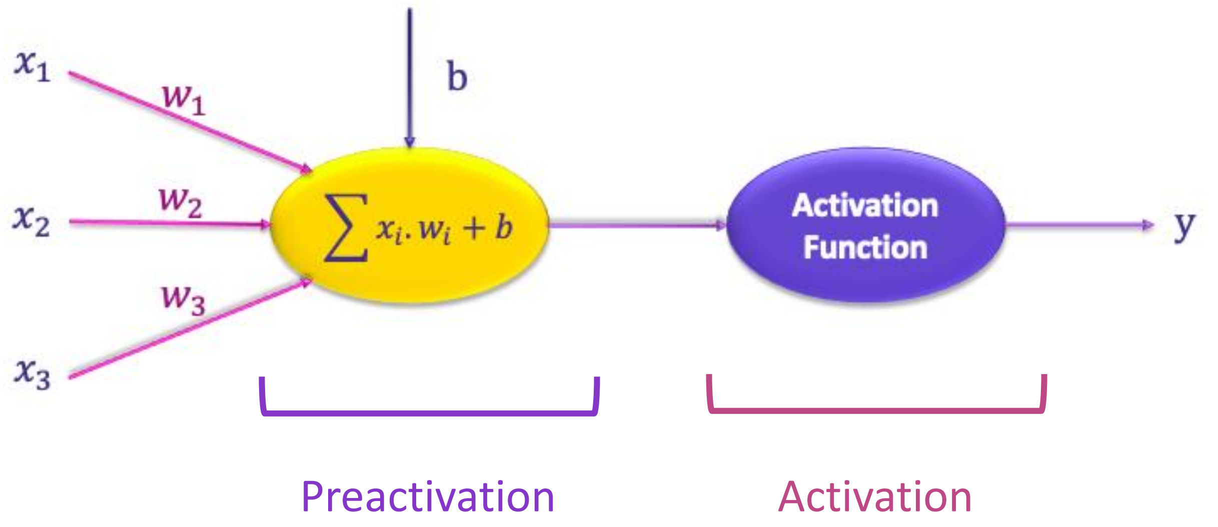
Forward Propagation

What are Feedforward Neural Networks?

At each neuron, two steps take place :

1. Preactivation : The weighted inputs are summed up

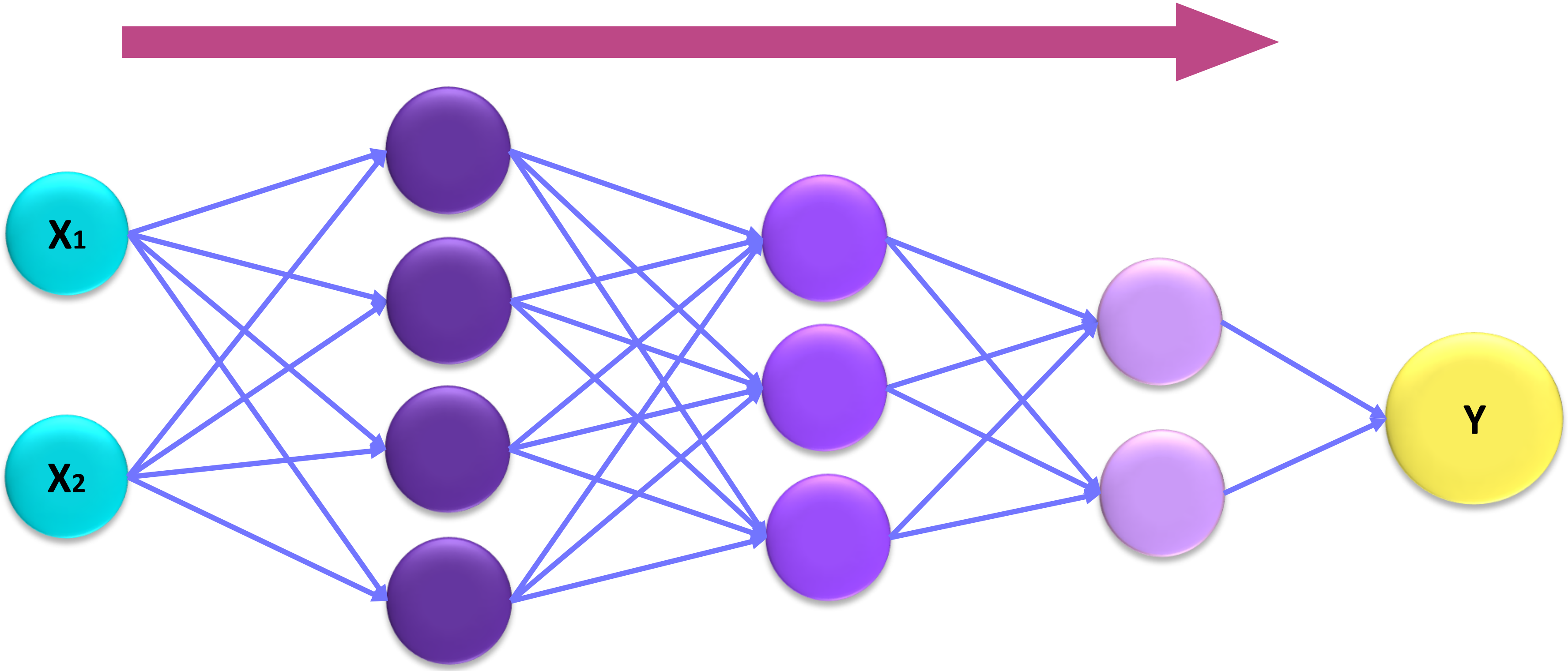
2. Activation : The weighted sum of inputs is passed to the activation function



Forward Propagation

What are Feedforward Neural Networks?

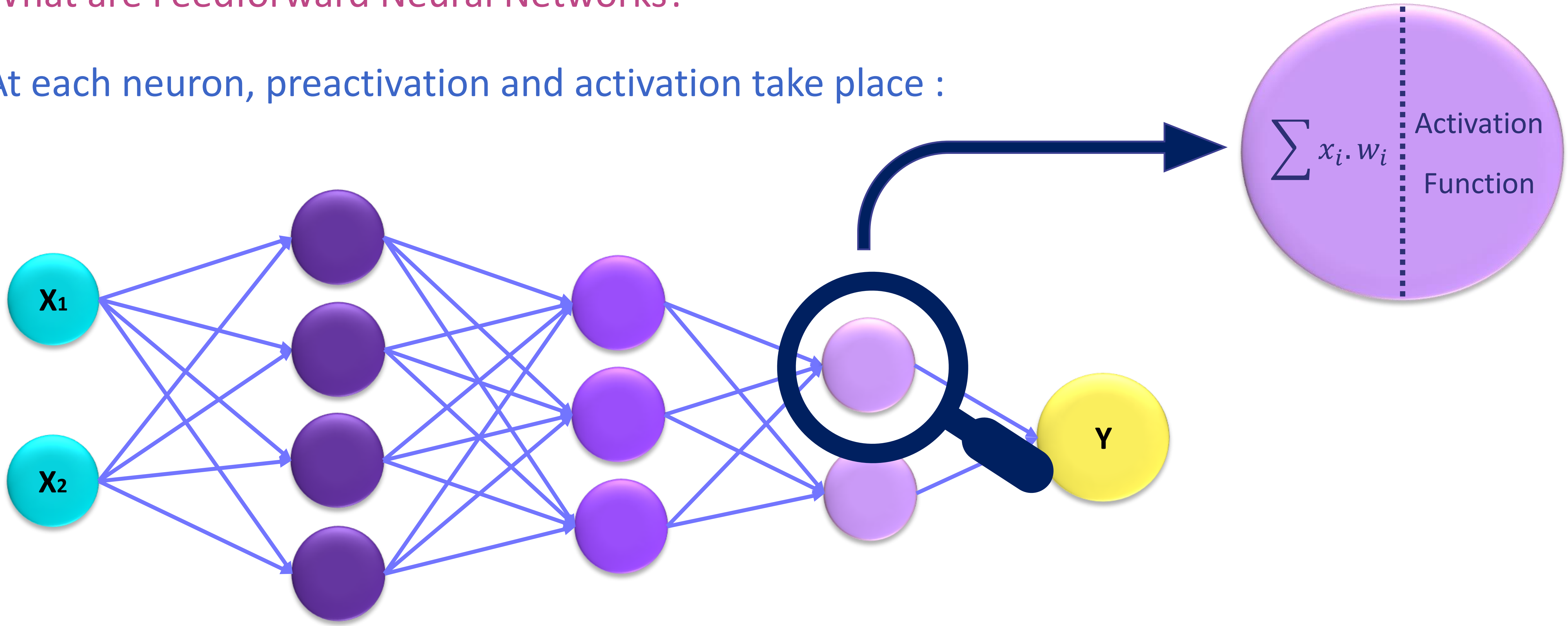
At each neuron, preactivation and activation take place :



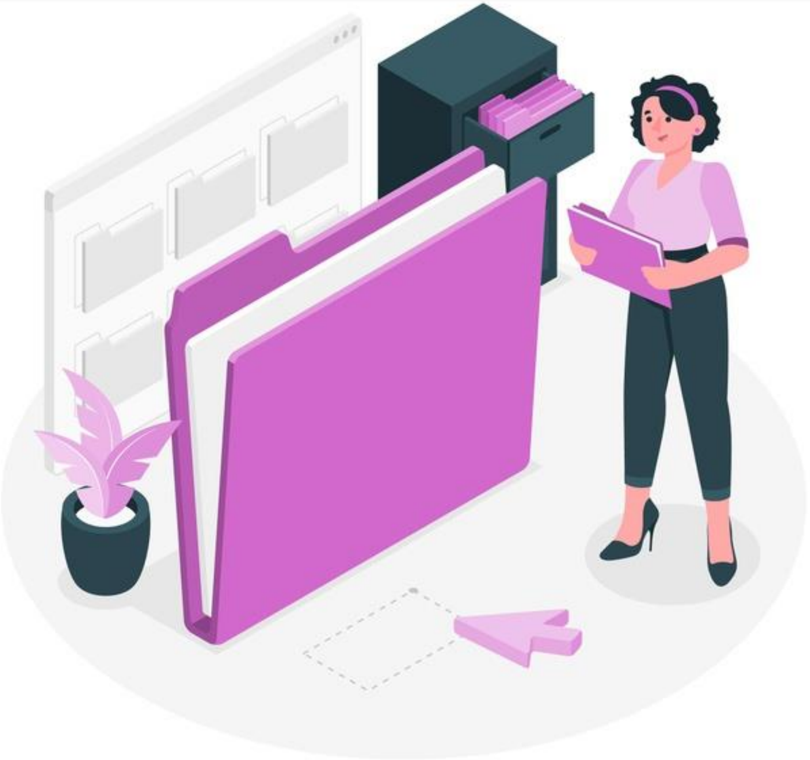
Forward Propagation

What are Feedforward Neural Networks?

At each neuron, preactivation and activation take place :



Classification and Regression



Classification and Regression

What is the difference between classification and regression ?

Classification and Regression

What is the difference between classification and regression ?

Classification

Outputs a label (class)

- Binary classification (2 classes)
  - cat
  - dog
- Multi-class classification (> 2 classes)
  - cat
  - dog
  - duck
  - hamster

Classification and Regression

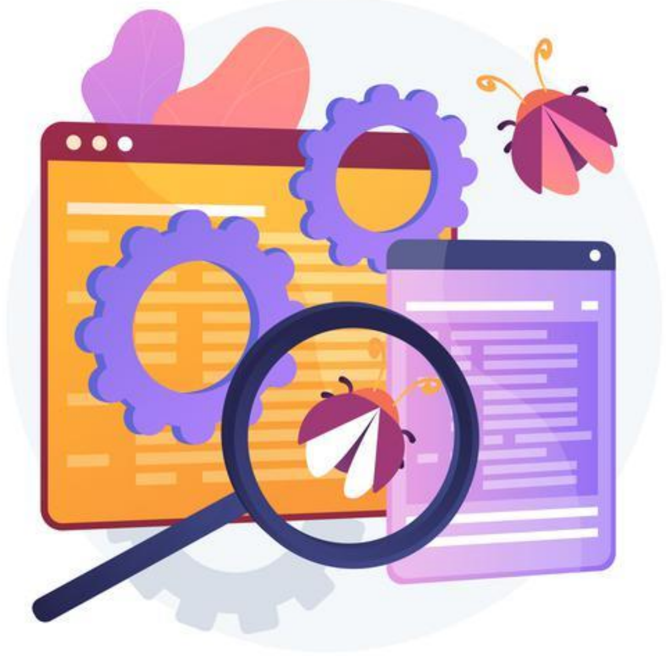
What is the difference between classification and regression ?

Regression

Predicts a quantity based on the learned data, such as :

- House price
- Weight
- Pandemic contamination rate

Cost Functions



Cost/Loss Functions

What is a cost / loss function ?

- The cost function measures the error between the true value and the predicted value by the trained model
- It evaluates how well the trained model performs
- If performance is bad, the cost function yield a high number

Cost/Loss Functions

What is a cost/loss function ?

- The better the performance, the lower the cost function result
- During training, the cost function result **must** decrease. Otherwise, this means that our model is not learning !
- Our aim is therefore to MINIMIZE the Cost function

Cost/Loss Functions

1 Regression Loss :

Mean Squared Error :

Performs direct comparisons between the true value and the model's output value

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{true_i} - y_{predicted_i})^2$$

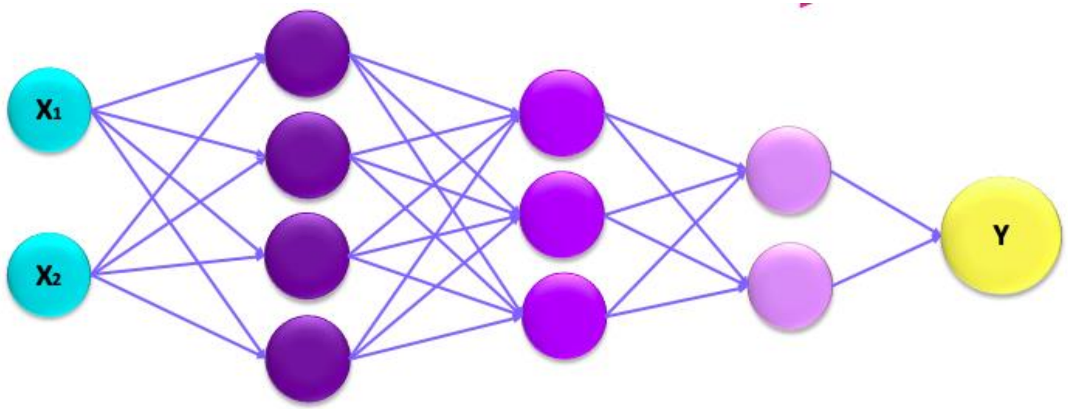
n = Total number of data points

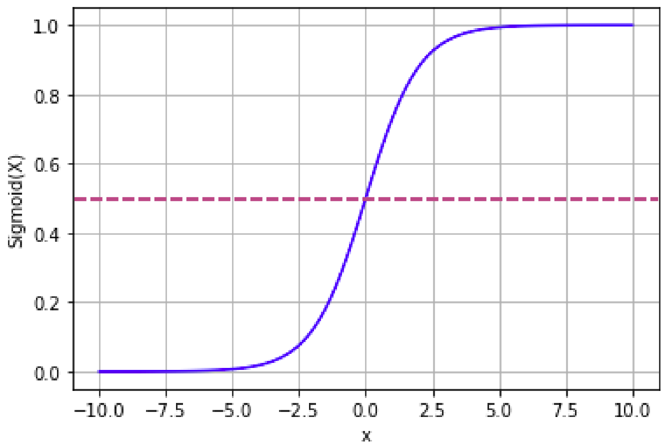
Cost/Loss Functions

2 Classification Losses :

Binary Classification Loss :

- Binary classification has 1 output node giving a probability (using the sigmoid activation function)





Probability = 0.5

Cost/Loss Functions

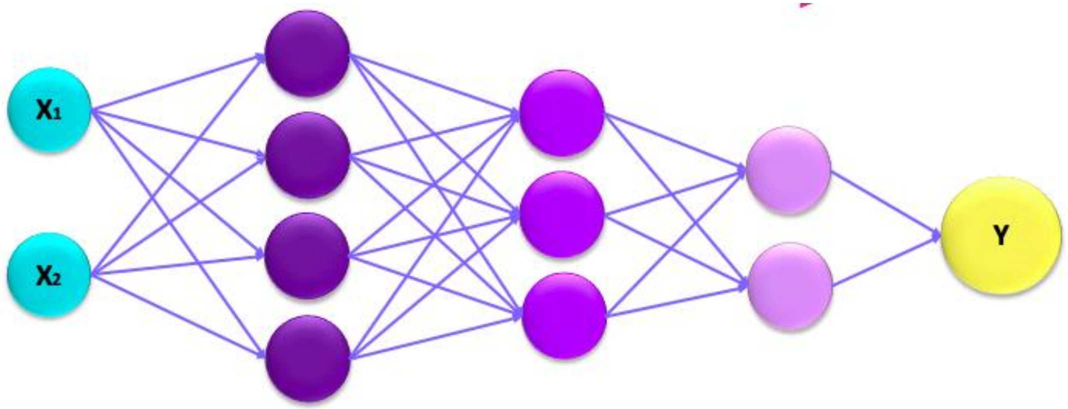
2 Classification Losses :

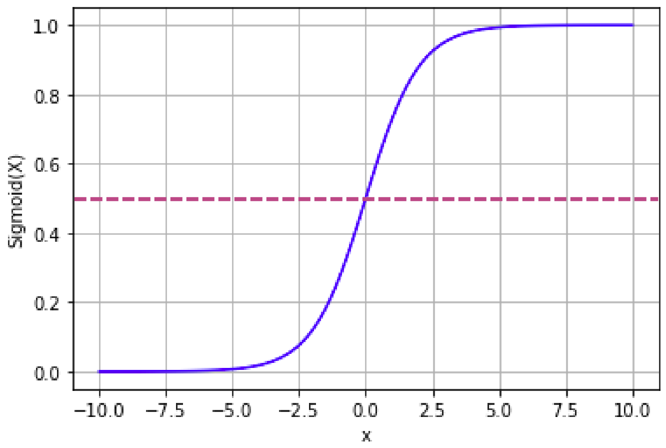
If probability ≥ 0.5 → label 1 (‘*dog*’)

If probability < 0.5 → label 0 (‘*cat*’)

Binary Classification Loss :

- Binary classification has 1 output node giving a probability (using the sigmoid activation function)





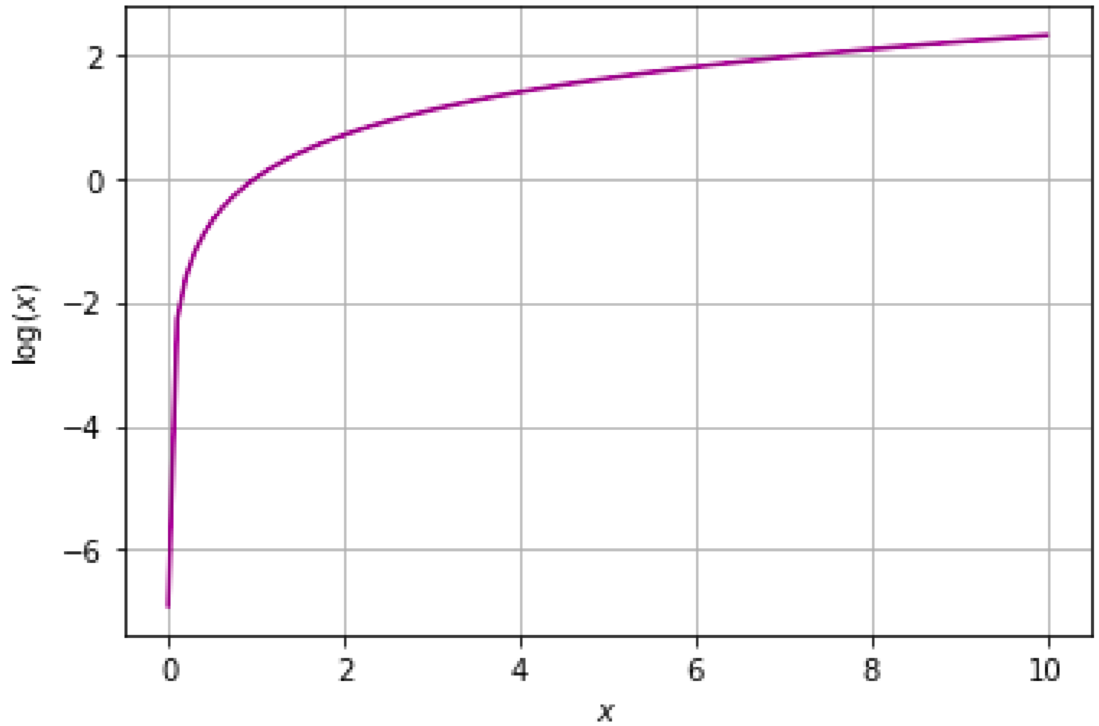
Probability = 0.5

Cost/Loss Functions

2 Classification Losses :

Let's review the Natural Logarithm (also known as *log to the base e* or *ln*)

Binary Classification Loss :



- As x approaches 0, $ln(x)$ approaches $-\infty$
- $ln(1) = 0$

Cost/Loss Functions

2

Classification Losses :

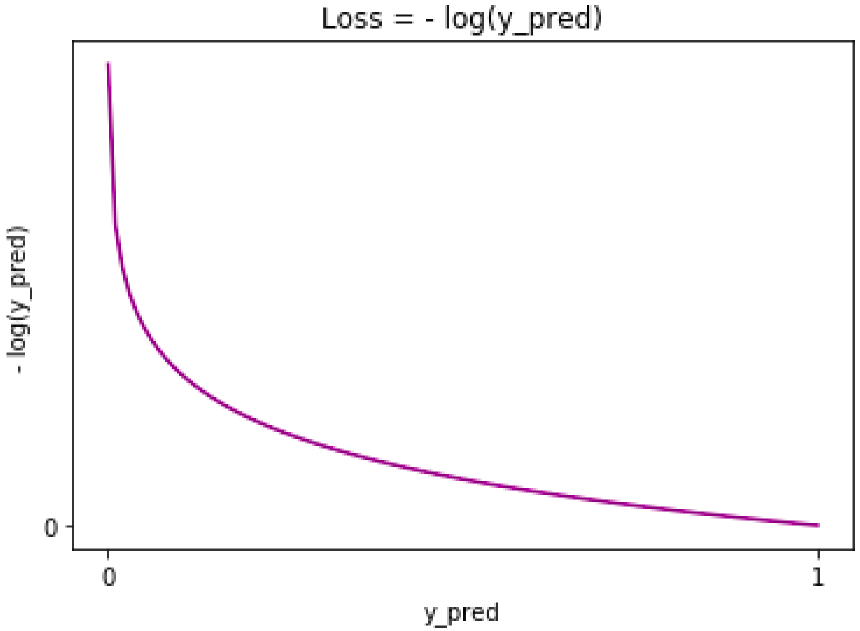
- Binary Classification Loss : The loss function used for binary classification is the **Binary Cross Entropy (BCE)**

Note : Binary **Cross Entropy Loss** is also known as **Log Loss**

Cost/Loss Functions

2 Classification Losses :

Binary Classification Loss :



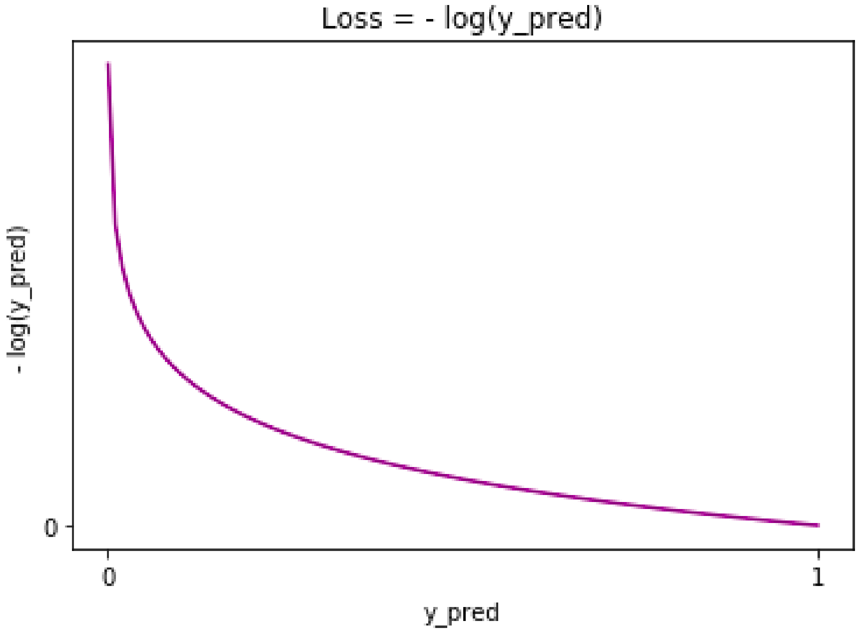
$$Loss = -log(y_{pred})$$

Used when we need to predict the class with label = 1 ($y_{true} = 1$)

Cost/Loss Functions

2 Classification Losses :

Binary Classification Loss :



Example:

y_{true}	y_{pred}
1	0.92
1	0.09

$$Loss = -log(y_{pred})$$

Used when we need to predict the class with label = 1 ($y_{true} = 1$)

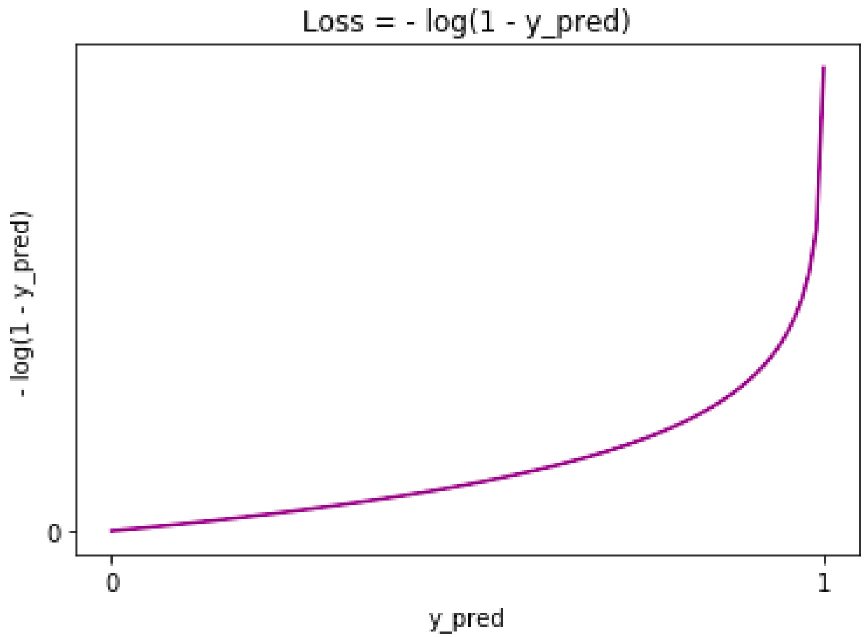
For $y_{pred} = 0.92$ ➔ Loss = - log(0.92) = 0.083 ✔

For $y_{pred} = 0.09$ ➔ Loss = - log(0.09) = 2.41

Cost/Loss Functions

2 Classification Losses :

Binary Classification Loss :



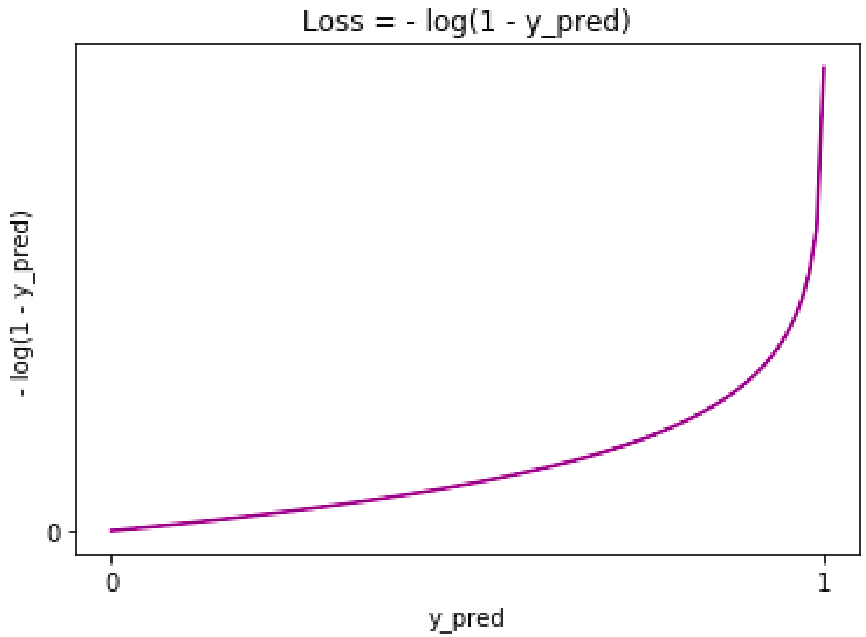
$$Loss = -\log(1 - y_{\text{pred}})$$

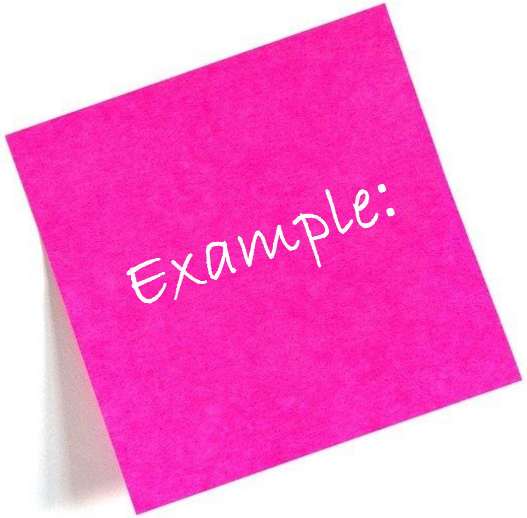
Used when we need to predict the class with label = 0 ($y_{true} = 0$)

Cost/Loss Functions

2 Classification Losses :

Binary Classification Loss :





y_{true}	y_{pred}
0	0.03
0	0.94

$$Loss = -\log(1 - y_{\text{pred}})$$

Used when we need to predict the class with label = 0 ($y_{true} = 0$)

For $y_{\text{pred}} = 0.03$ ➔ Loss = - log(1 – 0.03) = 0.03 ✔

For $y_{\text{pred}} = 0.94$ ➔ Loss = - log(1 – 0.94) = 2.81

Cost/Loss Functions

2 Classification Losses :

The loss function used for binary classification is the **Binary Cross Entropy (BCE)**

★ Binary Classification Loss :

The complete mathematical representation of the BCE loss function :

$$Loss = y_{true}\left(-\log(y_{pred})\right) + (1 - y_{true})(-\log(1 - y_{pred}))$$

Cost/Loss Functions

2 Classification Losses :

★ Multiclass Classification Loss : The loss function used for multiclass classification is the **Categorical Cross Entropy (CCE)**

Multiclass classification predicts 1 possible class out of several

Cost/Loss Functions

2 Classification Losses :

Multiclass Classification Loss :

Multiclass classification predicts 1 possible class out of several

What if we added a sigmoid activation function for each output ?

Cost/Loss Functions

2 Classification Losses :

Multiclass Classification Loss :

Multiclass classification predicts 1 possible class out of several

Adding a sigmoid activation function to every output would give a probability to every class, but the sum of those probabilities would not add up to 1 !

➔ **We use <u>softmax</u> activation function**

Cost/Loss Functions

2 Classification Losses :

Multiclass Classification Loss :

Multiclass classification predicts 1 possible class out of several

ANN Outputs

$$\begin{bmatrix} 2.1 \\ 3.2 \\ 1.5 \end{bmatrix} \longrightarrow$$

$$Softmax(x) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

$$\frac{8.17}{8.17 + 24.53 + 4.48}$$

$$\frac{24.53}{8.17 + 24.53 + 4.48}$$

$$\frac{4.48}{8.17 + 24.53 + 4.48}$$

Probabilities

$$\longrightarrow \begin{bmatrix} 0.22 \\ 0.66 \\ 0.12 \end{bmatrix}$$

Cost/Loss Functions

2 Classification Losses :

The loss function used for multiclass classification is the **Categorical Cross Entropy (CCE)**

Multiclass Classification Loss :

ANN Outputs

$$\begin{bmatrix} 2.1 \\ 3.2 \\ 1.5 \end{bmatrix} \rightarrow \quad Softmax(x) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad \begin{matrix} \frac{8.17}{8.17 + 24.53 + 4.48} \\ \frac{24.53}{8.17 + 24.53 + 4.48} \\ \frac{4.48}{8.17 + 24.53 + 4.48} \end{matrix} \rightarrow$$

y_{pred}

Probabilities

Find the Log losses

y_{true}

$$\begin{bmatrix} 0.22 \\ 0.66 \\ 0.12 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Total loss = sum of the log losses

$$Loss = \sum_{i=1}^{n} y_{true_i}(-\log(y_{pred_i})) + (1 - y_{true_i})(-\log(1 - y_{pred_i}))$$

Cost/Loss Functions

2 Classification Losses :

Multilabel Classification Loss :

Multilabel classification can predict several classes at the same time

Example : recognize the fruits in a fruit basket

Cost/Loss Functions

2 Classification Losses :

Multilabel Classification Loss :

Multilabel classification can predict several classes at the same time

Example : recognize the fruits in a fruit basket

Cost/Loss Functions

2 Classification Losses :

Multilabel Classification Loss :

- We **cannot use softmax** because it forces to have one best class
- We add **sigmoid** to every ouput node to predict the unique probability of each class
- To calculate the loss, we do the same as with multiclass classification : calculate the log loss for every node then sum the losses up