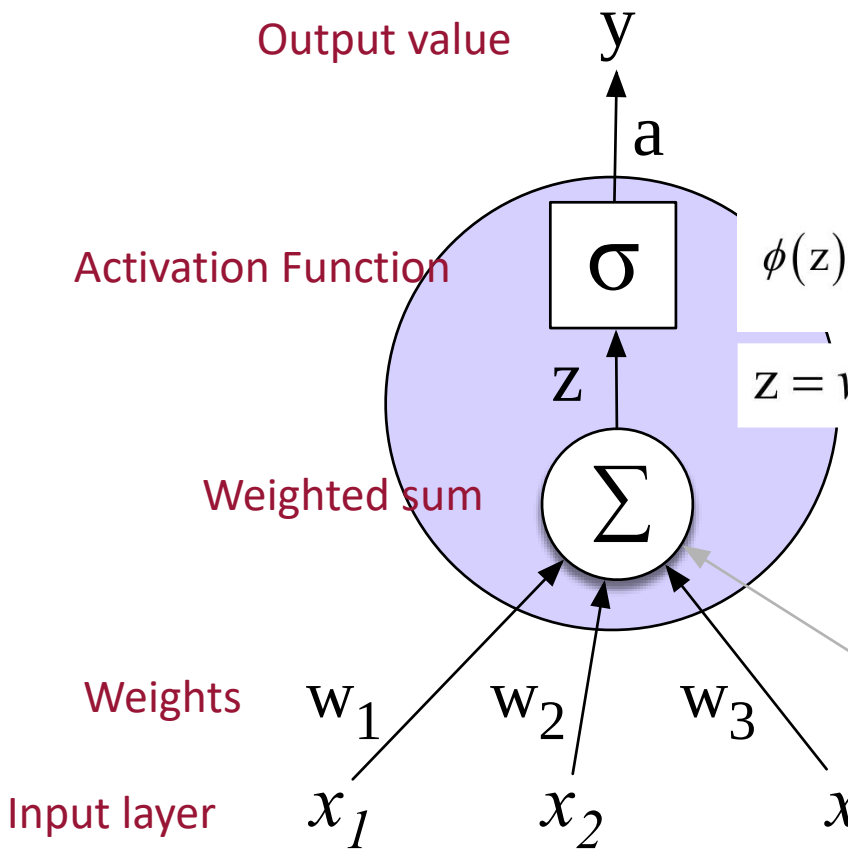
Activation Functions

Neural Network Unit

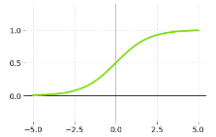
Perceptron



$$\phi(z) = \begin{cases} 1 \ \ if\ z \geq 0 \\ -1 \ \ otherwise \end{cases}$$

or

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



Sigmoid for binary classification

$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

Example

Suppose a unit has:

$w = [0.2, 0.3, 0.9]$

$b = 0.5$

What happens with input x:

$x = [0.5, 0.6, 0.1]$

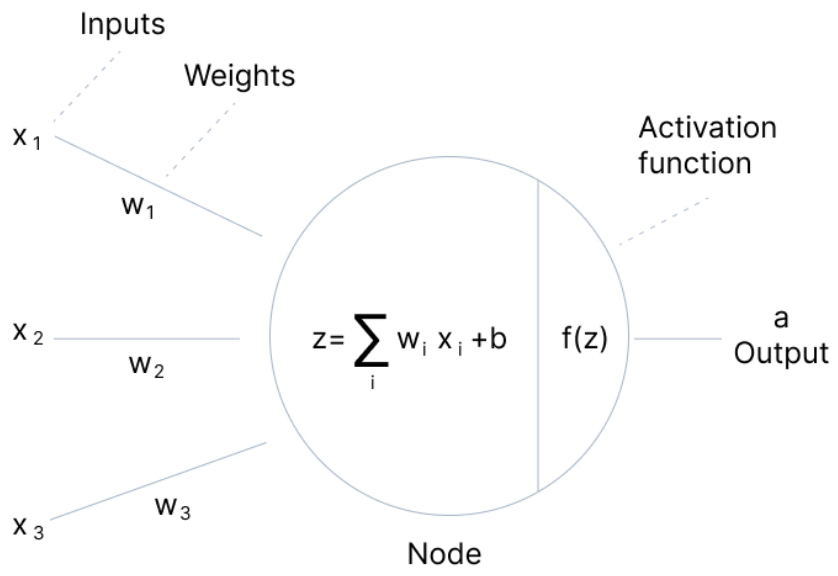
$$Z = w_1x_1 + w_2x_2 + w_3x_3 + b$$
$$= 0.2 \times 0.5 + 0.3 \times 0.6 + 0.9 \times 0.1 + 0.5$$
$$= 0.87$$

Output:
Using Step function
S(z)=1
Output= 1

If Sigmoid is taken with 0.5 as threshold
S(z) = 0.70
S(z)>=0.5 → 1

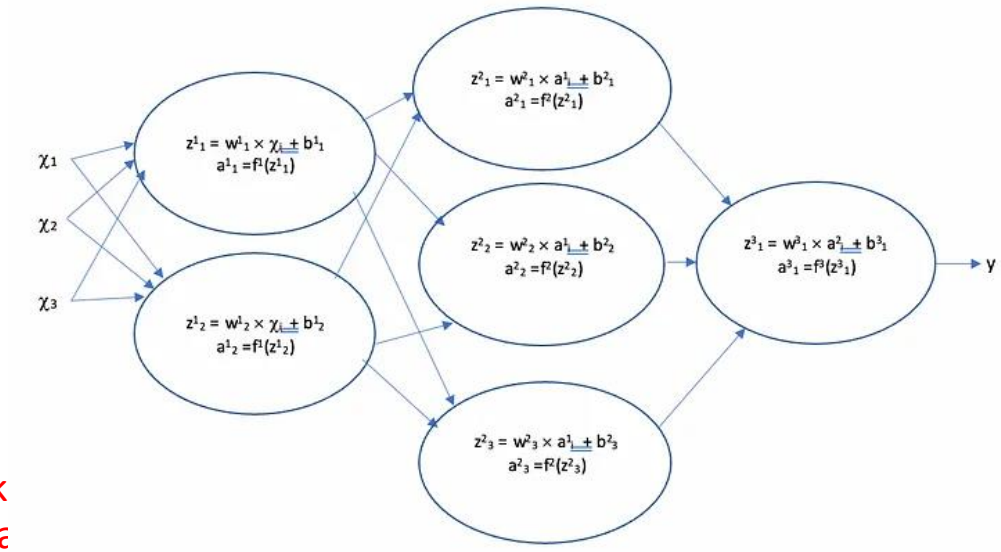
Activation functions

- An Activation Function decides whether a neuron should be activated (fired) or not.
- This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations.
- The activation function determines the output of the NN unit based on the weighted sum of the inputs and the bias term.
  - It contributes to the conversion of the input into a more usable output.
    - In other words it maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the chosen function)
  - The activation function may add non-linearity to the neural network.



Why need activation function?

- Activation functions introduce an additional step at each layer during the forward propagation.
- Let us suppose, we have a neural network working *without* the activation functions.
- In such case, the data would pass through the nodes and layers of the network only going through linear functions (w*x+b).
- Ie, every neuron will only be performing a linear transformation on the inputs using the weights and biases.
- Hence, no matter how many hidden layers we attach in the neural network all layers will behave in the same way because the composition of two linea functions is a linear function itself.
- Although the neural network will become simpler, learning any complex task is impossible, and our model would be just a linear regression model.



Types

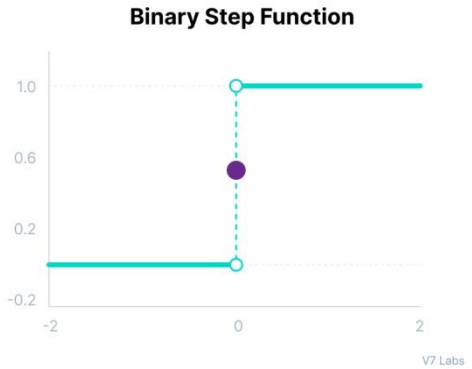
- Binary step Function
- Linear activation functions
- Non-linear activation functions

Activation Functions…

- Binary Step Function
  - Binary step function depends on a threshold value that decides whether a neuron should be activated or not.
    The input fed to the activation function is compared to a certain threshold; if the input is greater than it, then the neuron is activated, else it is deactivated, meaning that its output is not passed on to the next hidden layer.
  - Mathematically it can be represented as:

$$\textit{Binary step}$$

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



Limitations of binary step function:

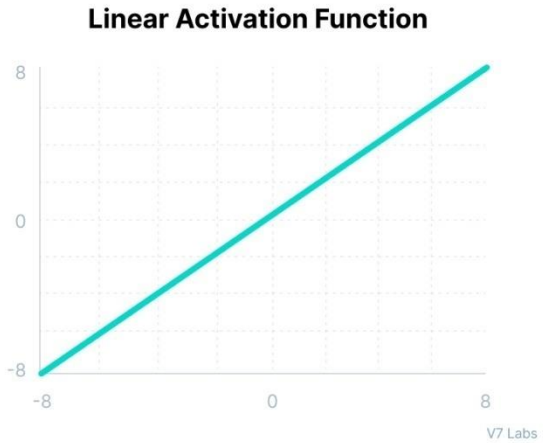
- It cannot provide multi-value outputs—hence, cannot be used for multi-class classification problems.
- The gradient of the step function is zero, which causes a hindrance in the backpropagation process.

Linear Activation Function

- The linear activation function, also known as "no activation," or "identity function" (multiplied x1.0), is where the activation is proportional to the input.
- The function doesn't do anything to the weighted sum of the input, it simply spits out the value it was given.

Mathematically it can be represented as:

$$f(x) = x$$



A linear activation function has two major problems :

- It’s not possible to use backpropagation as the derivative of the function is a constant and has no relation to the input x.
- All layers of the neural network will collapse into one if a linear activation function is used. No matter the number of layers in the neural network, the last layer will still be a linear function of the first layer. So, essentially, a linear activation function turns the neural network into just one layer.

Non-Linear Activation Functions

The linear activation function discussed so far is simply a linear regression model.

Because of its limited power, this does not allow the model to create complex mappings between the network’s inputs and outputs.

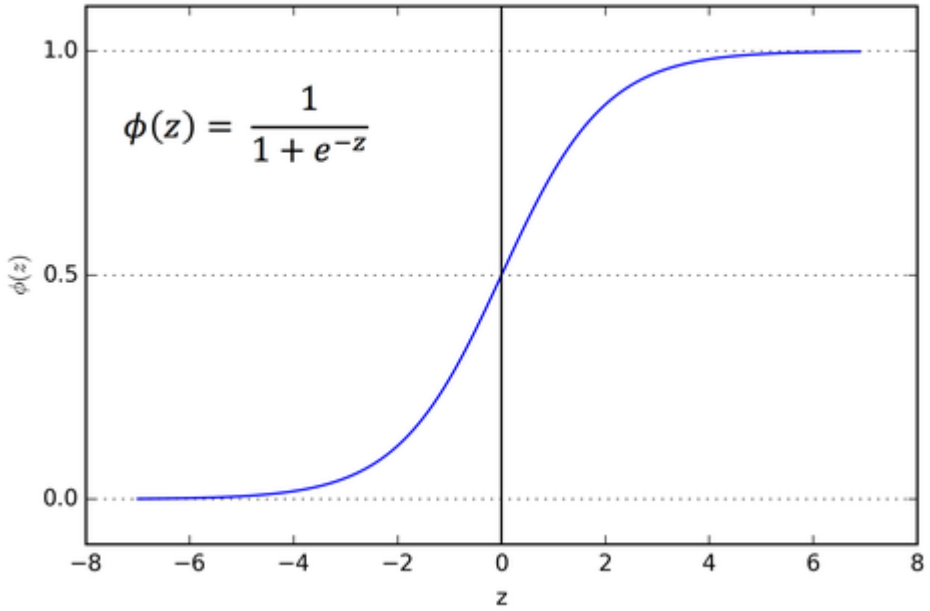
Non-linear Activation Functions

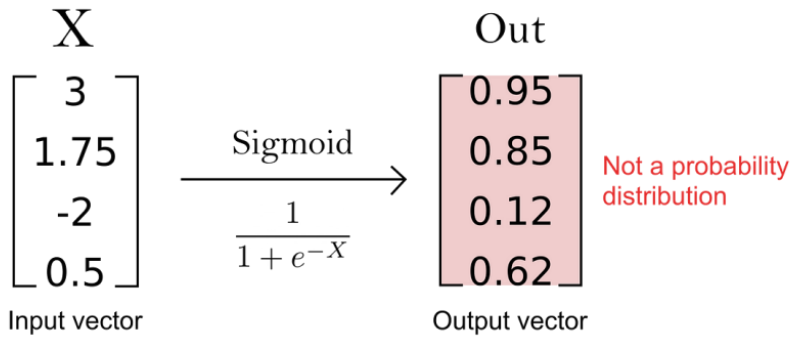
- Non-linear activation functions solve the following limitations of linear activation functions:
  - They allow backpropagation because now the derivative function would be related to the input, and it’s possible to go back and understand which weights in the input neurons can provide a better prediction.
  - They allow the stacking of multiple layers of neurons as the output would now be a non-linear combination of input passed through multiple layers. Any output can be represented as a functional computation in a neural network.

- The Nonlinear Activation Functions are mainly divided on the basis of their **range or curves**-

1. Sigmoid or Logistic Activation Function

- The Sigmoid Function curve looks like a S-shape.
- The main reason why we use sigmoid function is because it exists between (**0 to 1**).
- Therefore, it is especially used for models where we have to **predict the probability as an output**.
- Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.





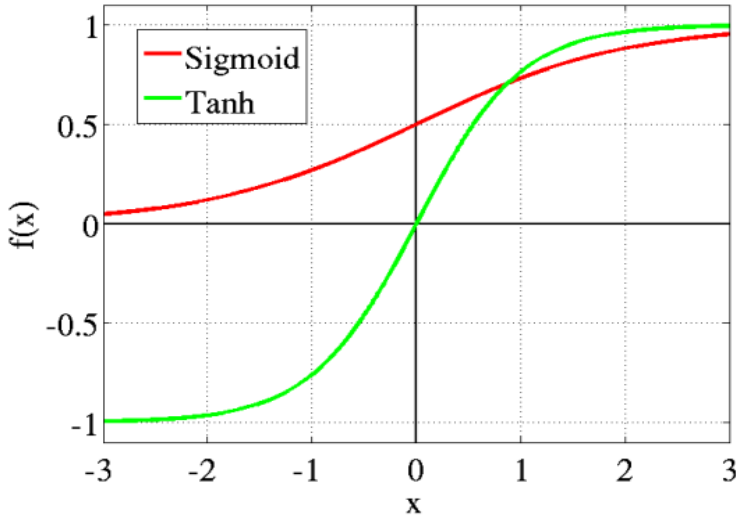
2. Tanh or hyperbolic tangent Activation Function

- Tanh is also like logistic sigmoid
- The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped).

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

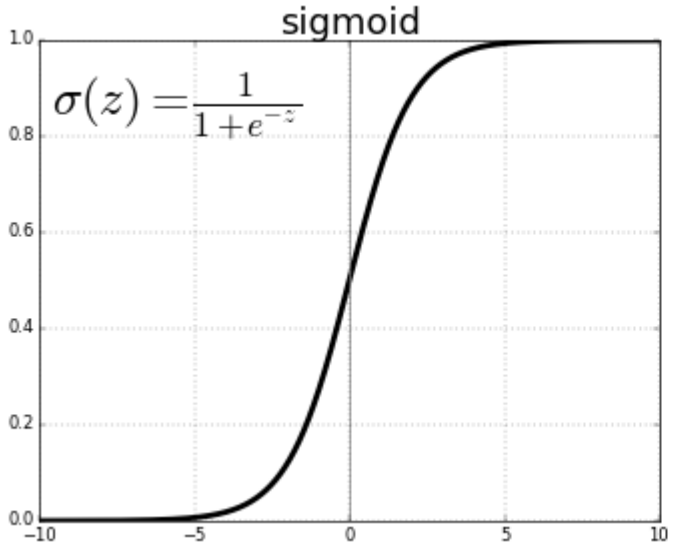
Where:

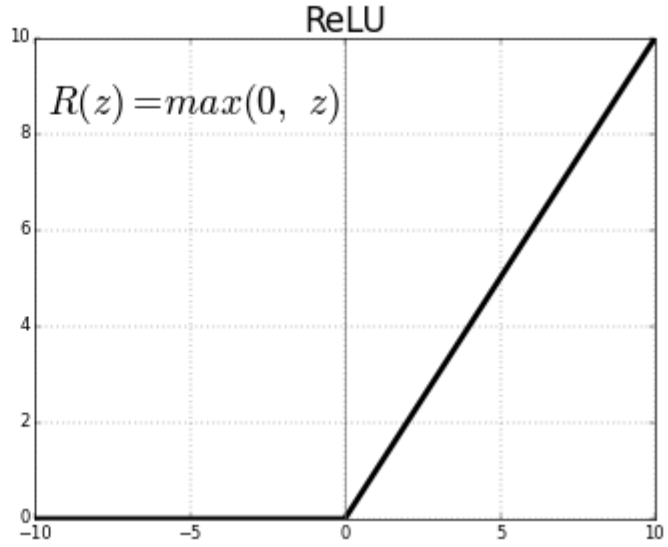
  - *x* represents the input value.
  - *e* is a mathematical constant approximately equal to **2.71828.**
- The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.



3. ReLU (Rectified Linear Unit) Activation Function

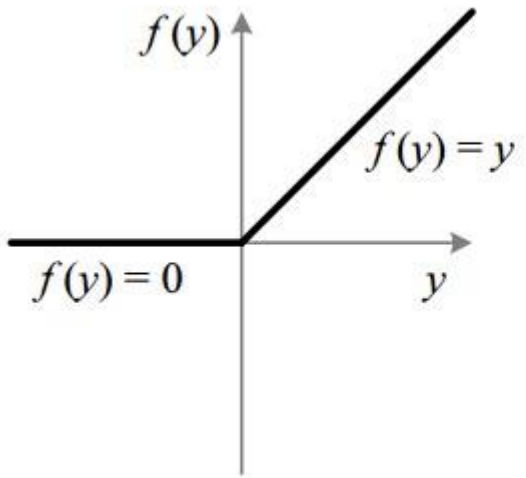
- The ReLU is one of the most used activation function in the world right now. Since, it is used in almost all the convolutional neural networks or deep learning.
- As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.
- But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly.
- That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately.

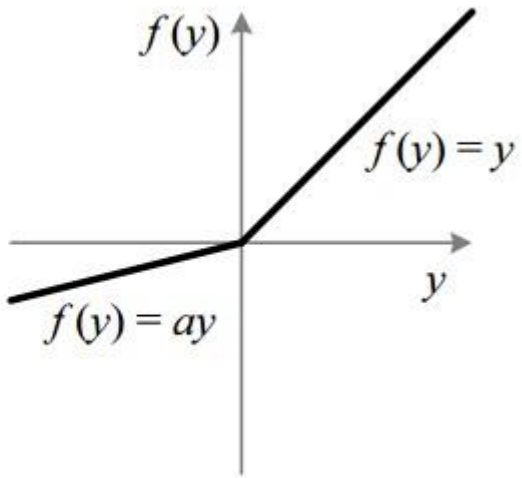




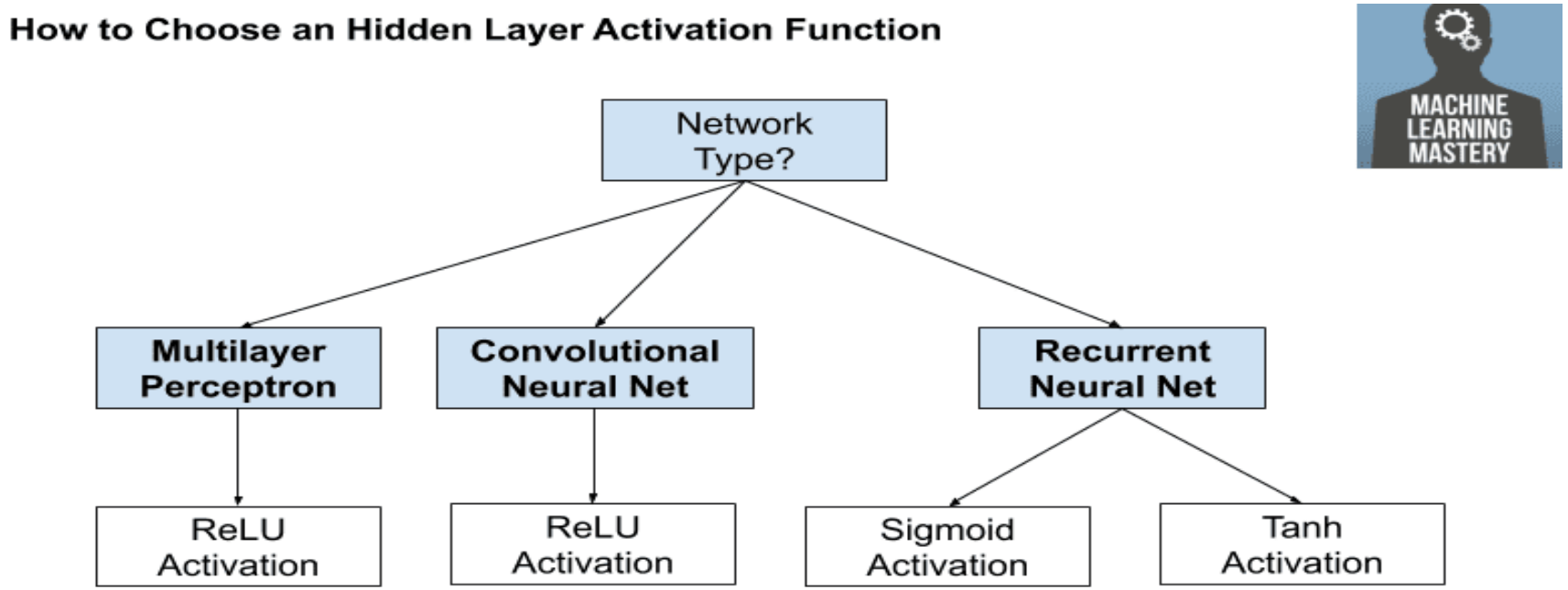
4. Leaky ReLU

- It is an attempt to solve the dying ReLU problem
- The leak helps to increase the range of the ReLU function. Usually, the value of **a** is 0.01 or so.
- When **a is not 0.01** then it is called **Randomized ReLU**





- Hidden layer activation functions:
  - Rectified Linear Activation (**ReLU**)
  - Logistic (**Sigmoid**)
  - Hyperbolic Tangent (**Tanh**)

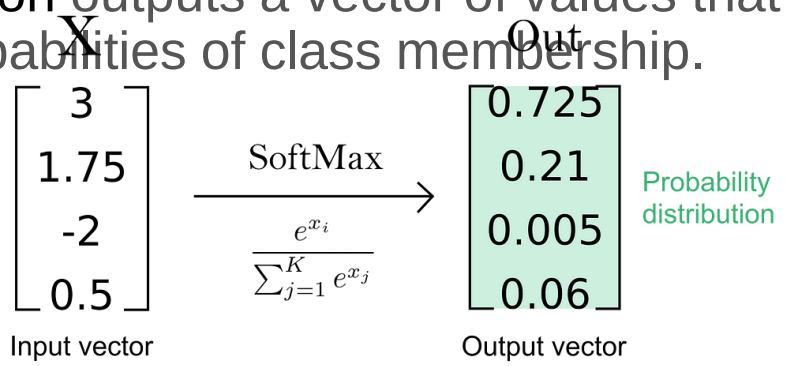


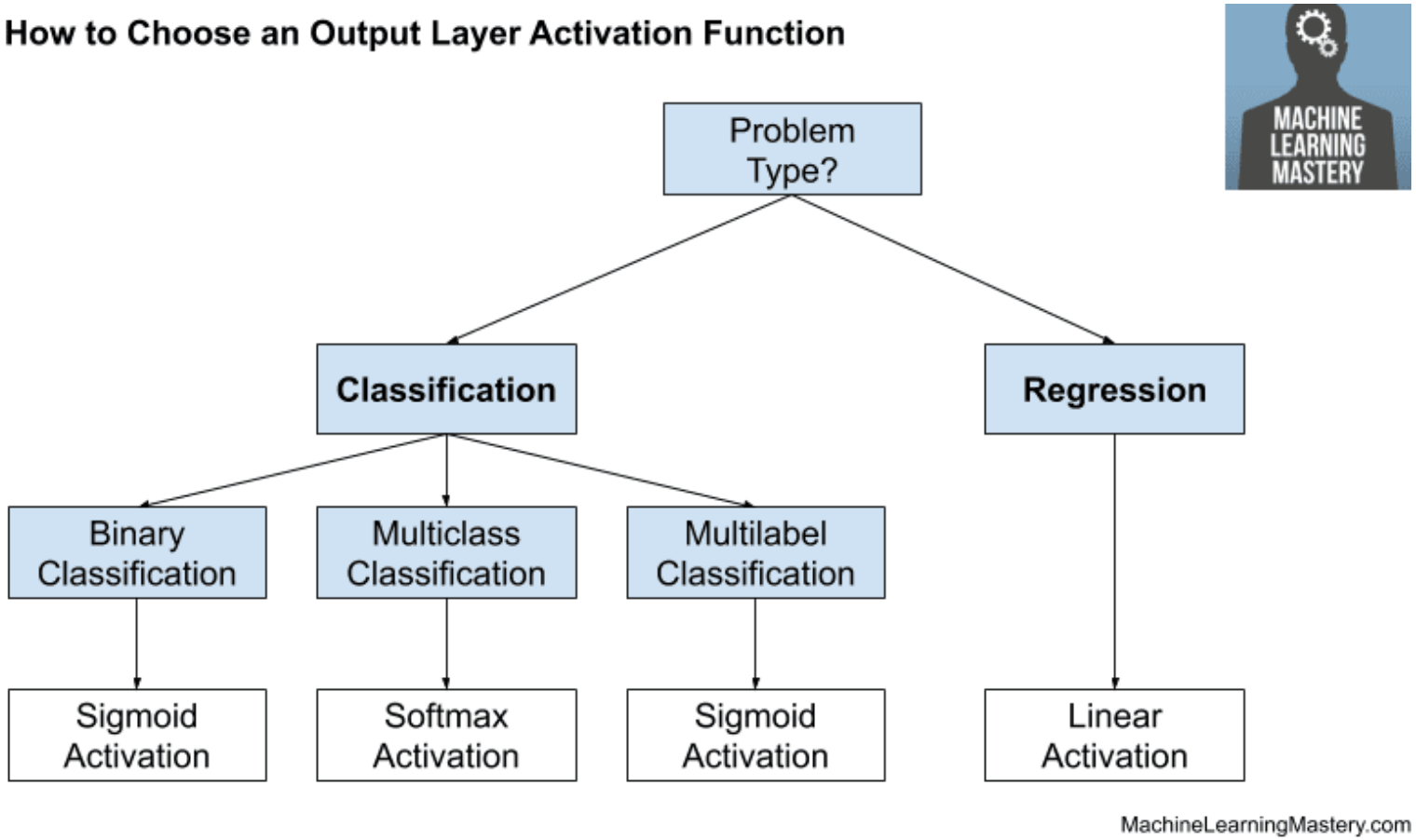
- **Activation functions for Output Layers**
  - Linear
  - Logistic (Sigmoid)
  - Softmax

- **Softmax**
  - For **Binary classification**, both **sigmoid**, as well as **softmax**, are equally approachable but in case of multi-class classification problem we generally use softmax
  - The softmax function is calculated as follows:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

  - The softmax function outputs a vector of values that sum to 1.0 that can be interpreted as probabilities of class membership.



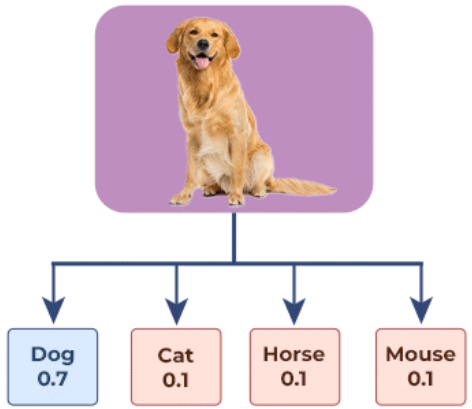


Multiclass classification :
The goal is to assign instances to one of multiple predefined classes or categories, where each instance belongs to exactly one class.

Multilabel classification is a machine learning task where each instance can be associated with multiple labels simultaneously

Mutliclass Classification vs multilabel classification

Multiclass Classification

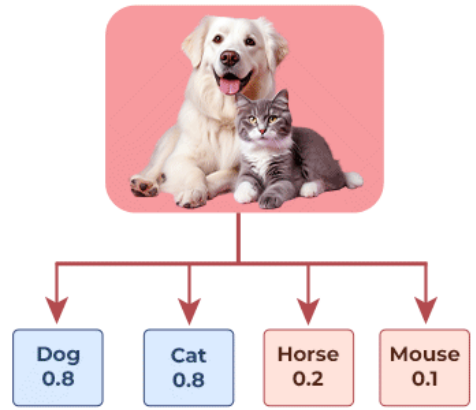


Classes

(pick one class)

- [x] Dog
- [ ] Cat
- [ ] Horse
- [ ] Mouse

Multilabel Classification



Classes

(pick all the labels present in the image)

- [x] Dog
- [x] Cat
- [ ] Horse
- [ ] Mouse