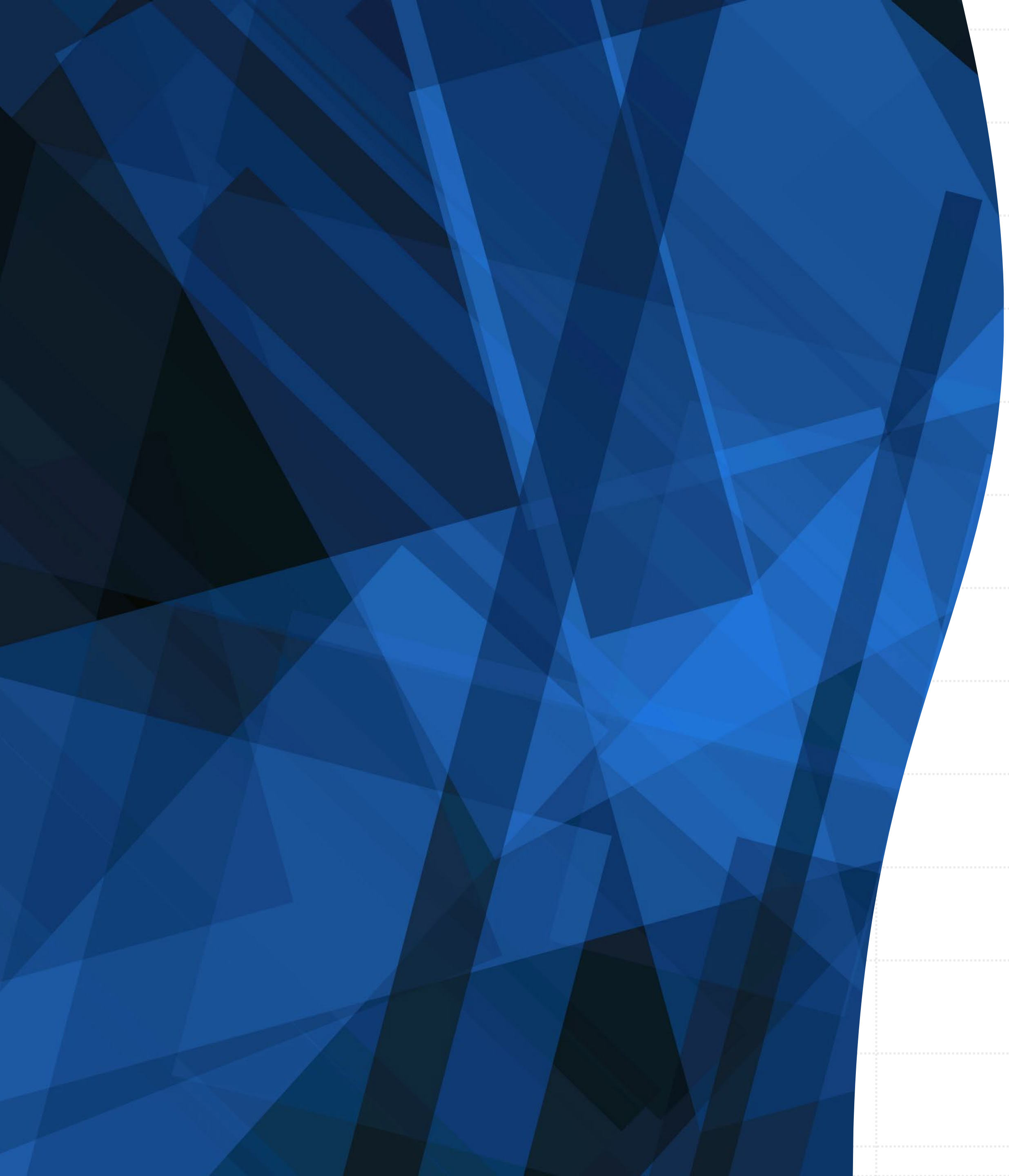


Neural network

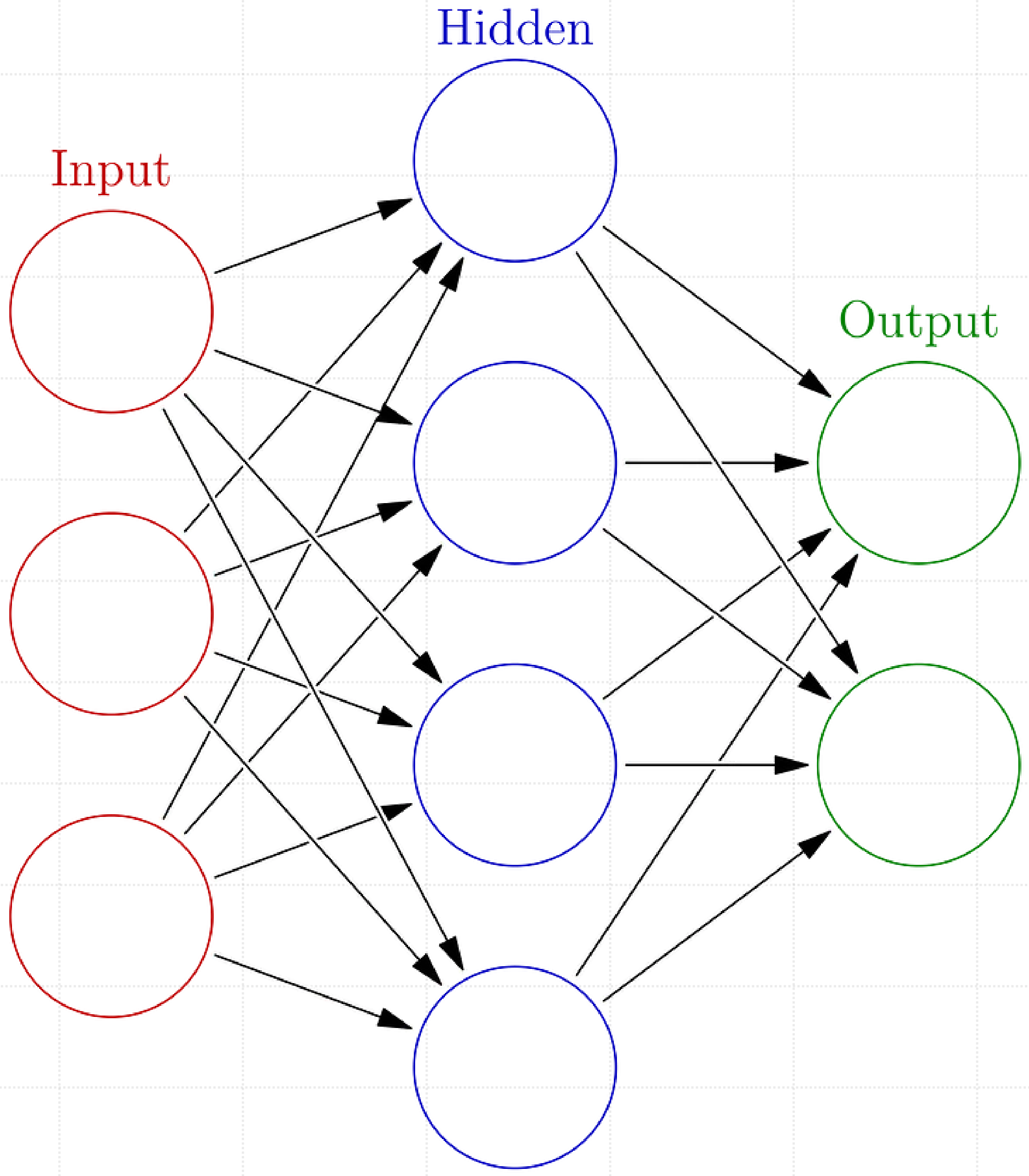
a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain

working

Brain – neurons – networks – electrical signals

ANN - made of artificial neurons that work together to solve a problem

structure



Input layer

Information from the outside world enters the artificial neural network from the input layer.

Input nodes process the data, analyze or categorize it, and pass it on to the next layer.

Hidden Layer

Hidden layers take their input from the input layer or other hidden layers.

Artificial neural networks can have a large number of hidden layers.

Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.

Output layer

The output layer gives the final result of all the data processing by the artificial neural network.

It can have single or multiple nodes.

For instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0.

However, if we have a multi-class classification problem, the output layer might consist of more than one output node.

Email classification example

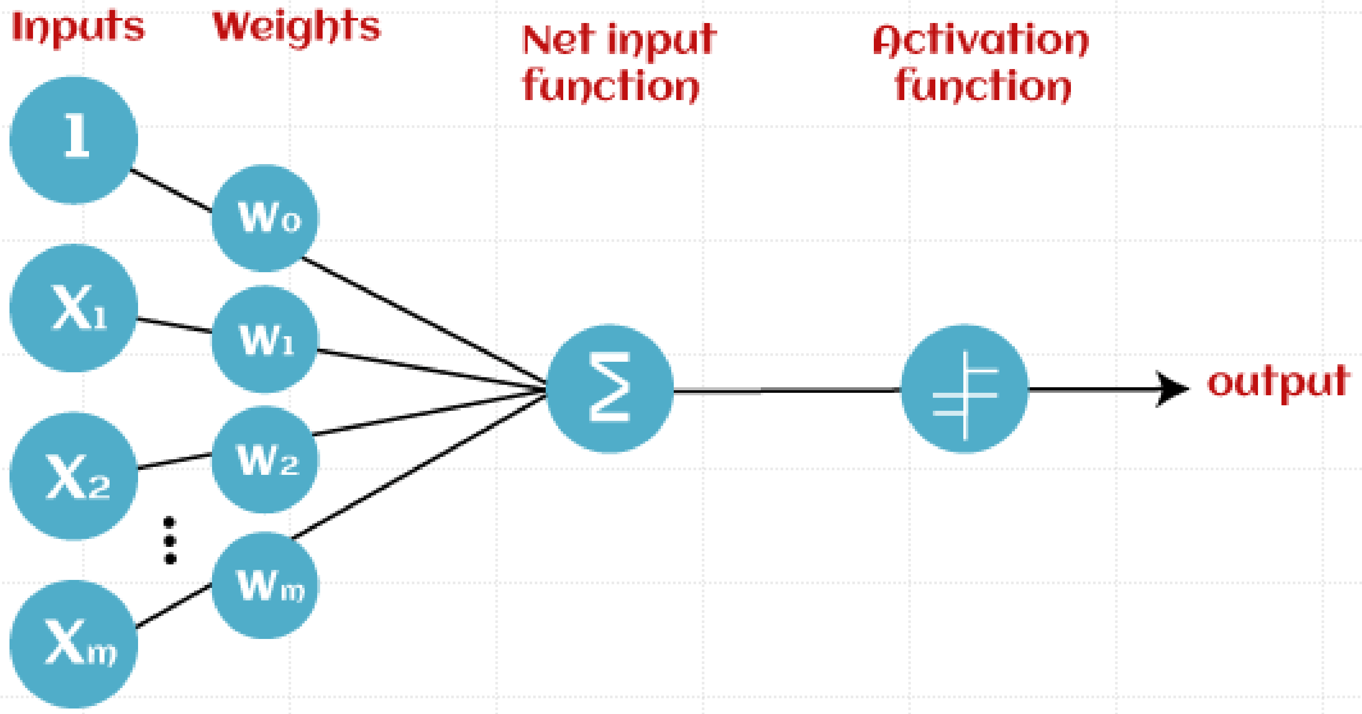
The input layer takes features like email content, sender information, and subject.

These inputs, multiplied by adjusted weights, pass through hidden layers

The network, through training, learns to recognize patterns indicating whether an email is spam or not.

The output layer, with a binary activation function, predicts whether the email is spam (1) or not (0).

network iteratively refines its weights through backpropagation


Inputs
Weights
Net input function
Activation function
1
X1
X2
Xm
W0
W1
W2
Wm
Σ
output

Two stages

Back propagation

Forward propagation

Forward Propagation

- **Input Layer:** Each feature in the input layer is represented by a node on the network, which receives input data
- **Weights and Connections**: The weight of each neuronal connection indicates how strong the connection is. Throughout training, these weights are changed.
- **Hidden Layers:** Each hidden layer neuron processes inputs by multiplying them by weights, adding them up, and then passing them through an activation function. By doing this, non-linearity is introduced, enabling the network to recognize intricate patterns
- Sign function
- Step function
- Sigmoid function
- **Output:** The final result is produced by repeating the process until the output layer is reached.

Back propagation

- **Loss Calculation**: The network’s output is evaluated against the real goal values, and a loss function is used to compute the difference.
- For a regression problem, the Mean Squared Error (MSE) is commonly used as the cost function.
- Loss Function:

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Back propagation

Gradient Descent: Gradient descent is then used by the network to reduce the loss.

To lower the inaccuracy, weights are changed based on the derivative of the loss with respect to each weight**.**

Back propagation

Adjusting weights: The weights are adjusted at each connection by applying this iterative process, or backpropagation, backward across the network

Training: During training with different data samples, the entire process of forward propagation, loss calculation, and backpropagation is done iteratively, enabling the network to adapt and learn patterns from the data.

Actvation Functions: Model non-linearity is introduced by activation functions like the rectified linear unit (ReLU) or sigmoid. Their decision on whether to “fire” a neuron is based on the whole weighted input.

Types of neural network

Feed forward neural network : data moves from input to output in one single direction → no feedback loop → straight forward architecture → appropriate for (regression and pattern recognition)

Multilayer Perceptron (MLP): MLP is a type of feedforward neural network with three or more layers, including an input layer, one or more hidden layers, and an output layer. It uses nonlinear activation functions.

Types of neural network

Convolutional Neural Network (CNN): A CNN is a specialized artificial neural network designed for image processing.

It employs convolutional layers to automatically learn hierarchical features from input images, enabling effective image recognition and classification.

CNNs have revolutionized computer vision and are pivotal in tasks like object detection and image analysis.

Types of Neural network

- Recurrent Neural Network (RNN): An artificial neural network type intended for sequential data processing is called a Recurrent Neural Network (RNN).
- It is appropriate for applications where contextual dependencies are critical, such as time series prediction and natural language processing, since it makes use of feedback loops, which enable information to survive within the network.

Types of neural network

Long Short-Term Memory (LSTM): LSTM is a type of RNN that is designed to overcome the vanishing gradient problem in training RNNs.

It uses memory cells and gates to selectively read, write, and erase information.

perceptron

Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence.

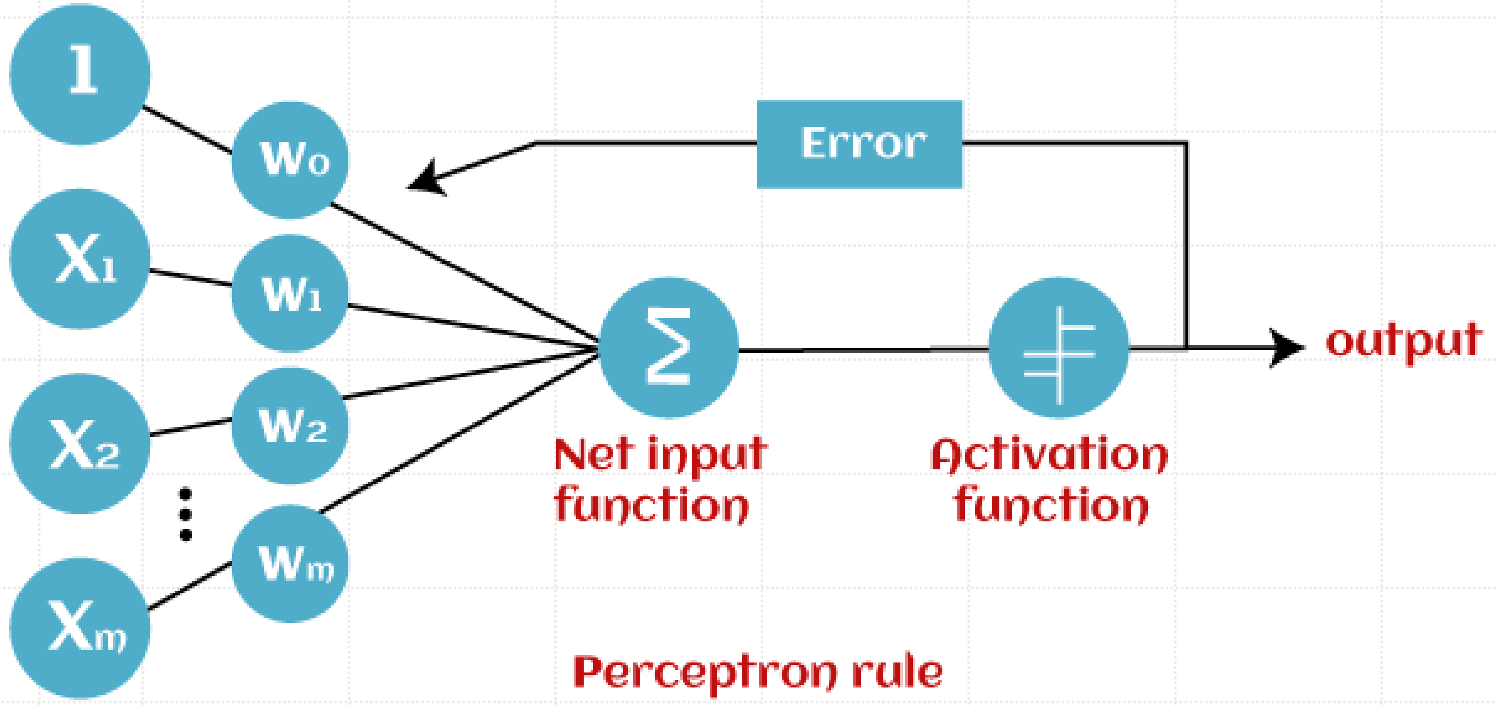
Perceptron model is also treated as one of the best and simplest types of Artificial Neural networks.

Perceptron

It is a supervised learning algorithm of binary classifiers.

Hence, we can consider it as a single-layer neural network with four main parameters

- **input values**
- **weights and Bias**
- **net sum**
- **activation function.**



Activation function also called step function (f)

ensuring that output is mapped between required values (0,1) or (-1,1)

steps

first step first, multiply all input values with corresponding weight values and then add them to determine the weighted sum. Mathematically, we can calculate the weighted sum as follows:

$\sum wi*xi = x1*w1 + x2*w2 +\dots wn*xn$

Add bias

Add a special term called **bias 'b'** to this weighted sum to improve the model's performance.

$\sum \mathbf{wi*xi + b}$

activation function f is applied with the above-mentioned weighted sum, which gives us output either in binary form or a continuous value as follows:

$$Y = f(\sum wi*xi + b)$$

Single layer perceptron

- A single-layered perceptron model consists feed-forward network and also includes a threshold transfer function inside the model.
- The main objective of the single-layer perceptron model is to analyze the linearly separable objects with binary outcomes.
- In a single layer perceptron model, its algorithms do not contain recorded data, so it begins with inconstantly allocated input for weight parameters.
- Further, it sums up all inputs (weight). After adding all inputs, if the total sum of all inputs is more than a predetermined value, the model gets activated and shows the output value as +1.

Single layer perceptron

- If the outcome is same as pre-determined or threshold value, then the performance of this model is stated as satisfied, and weight demand does not change.
- However, this model consists of a few discrepancies triggered when multiple weight inputs values are fed into the model.
- Hence, to find desired output and minimize errors, some changes should be necessary for the weights input.
- "Single-layer perceptron can learn only linearly separable patterns."

Multi-Layered Perceptron Model

Forward stage

Backward stage

Perceptron function

f(x)=1; if w.x+b>0

otherwise, f(x)=0

'w' represents real-valued weights vector

'b' represents the bias

'x' represents a vector of input x values.

perceptron model has the following characteristics.

Perceptron is a machine learning algorithm for supervised learning of binary classifiers

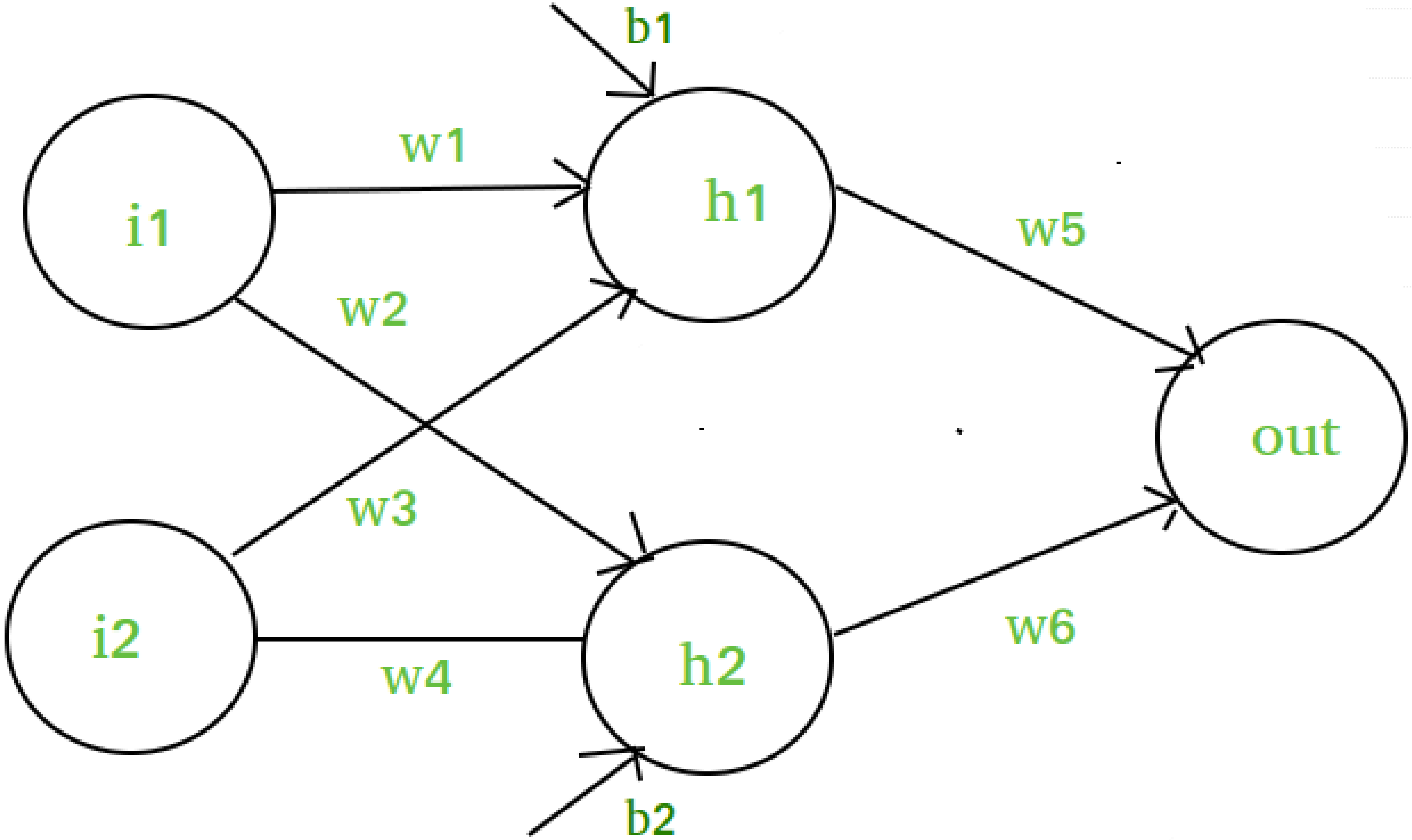
Perceptron the weight coefficient is automatically learned.

Initially, weights are multiplied with input features, and the decision is made whether the neuron is fired or not.

The activation function applies a step rule to check whether the weight function is greater than zero.

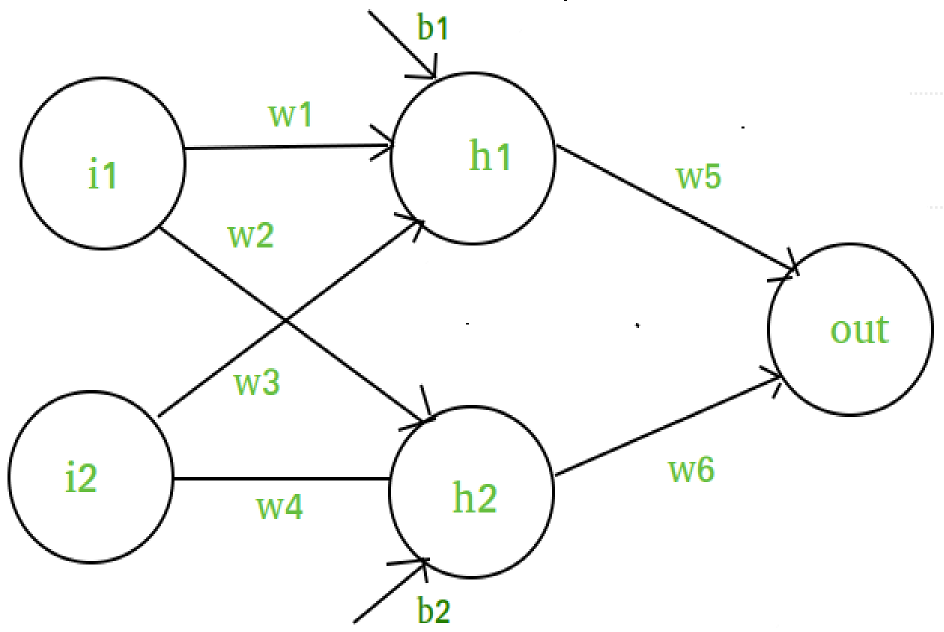
The linear decision boundary is drawn, enabling the distinction between the two linearly separable classes +1 and -1

If the added sum of all input values is more than the threshold value, it must have an output signal; otherwise, no output will be shown


b1
w1
i1
h1
w5
w2
out
w3
i2
h2
w6
w4
b2

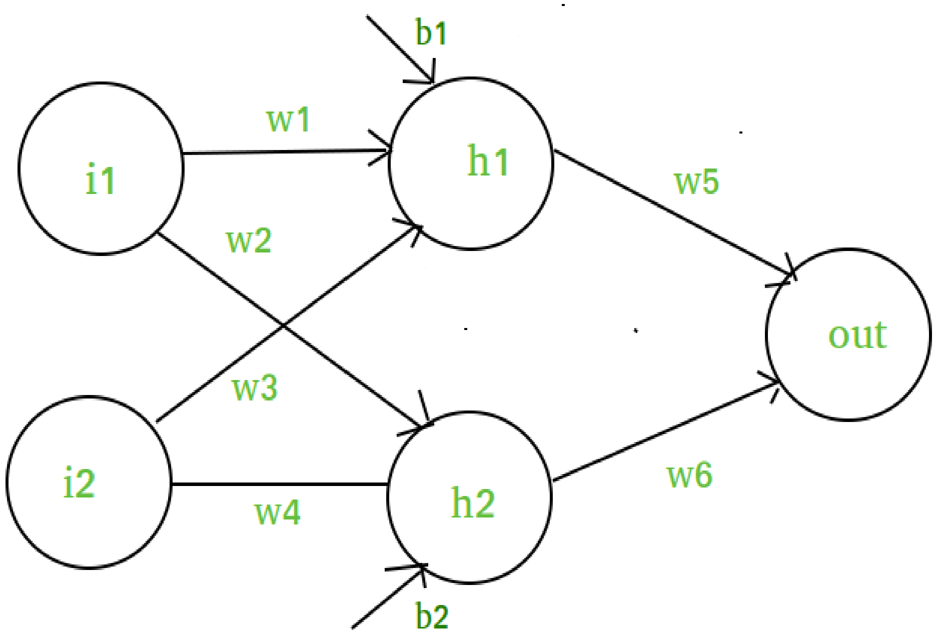
Hidden layer → layer 1

- z(1) = W(1)X + b(1) a(1)



- z(1) is the vectorized output of layer 1
- W(1) be the vectorized weights assigned to neurons of hidden layer i.e. *w1, w2, w3 and w4*
- X be the vectorized input features i.e. *i1 and i2*
- b is the vectorized bias assigned to neurons in hidden layer i.e. *b1 and b2*
- a(1) is the vectorized form of any linear function.

Output layer

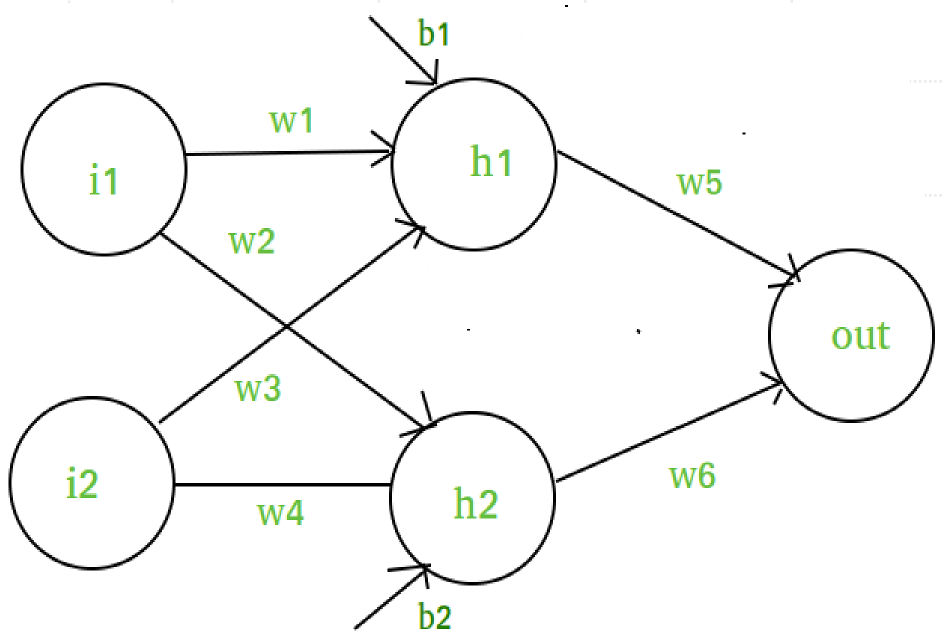


Input for layer 2 is output from layer 1

z(2) = W(2)a(1) + b(2)

a(2) = z(2)

Calculation at output layer



z(2) = (W(2) * [W(1)X + b(1)]) + b(2)

z(2) = [W(2) * W(1)] * X + [W(2)*b(1) + b(2)]

Let,

[W(2) * W(1)] = W

[W(2)*b(1) + b(2)] = b

Final output : z(2) = W*X + b

which is again a linear function

the composition of two linear function is a linear function itself

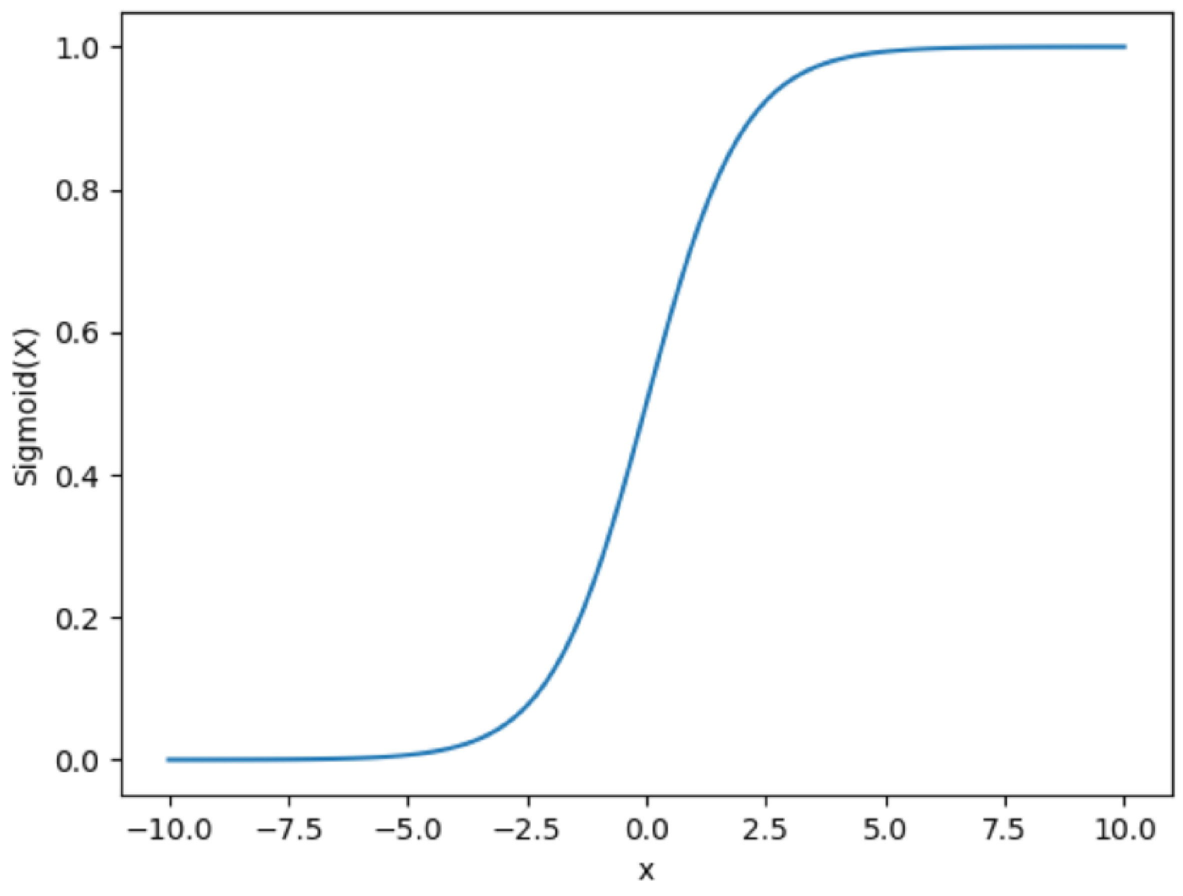
Linear function

- **Linear Function**
- **Equation :** Linear function has the equation similar to as of a straight line i.e. **y = x**
- Problem : If we will differentiate linear function to bring non-linearity, result will no more depend on *input “x”* and function will become constant, it won’t introduce any ground-breaking behavior to our algorithm.

Sigmoid Function

- It is a function which is plotted as **'S'** shaped graph.
- **Equation :** $A = 1/(1 + e^{-x})$
- **Nature :** Non-linear. Notice that X values lies between -2 to 2, Y values are very steep. This means, small changes in x would also bring about large changes in the value of Y.
- **Value Range :** 0 to 1
- **Uses :** Usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be ***1*** if value is greater than **0.5** and ***0*** otherwise.

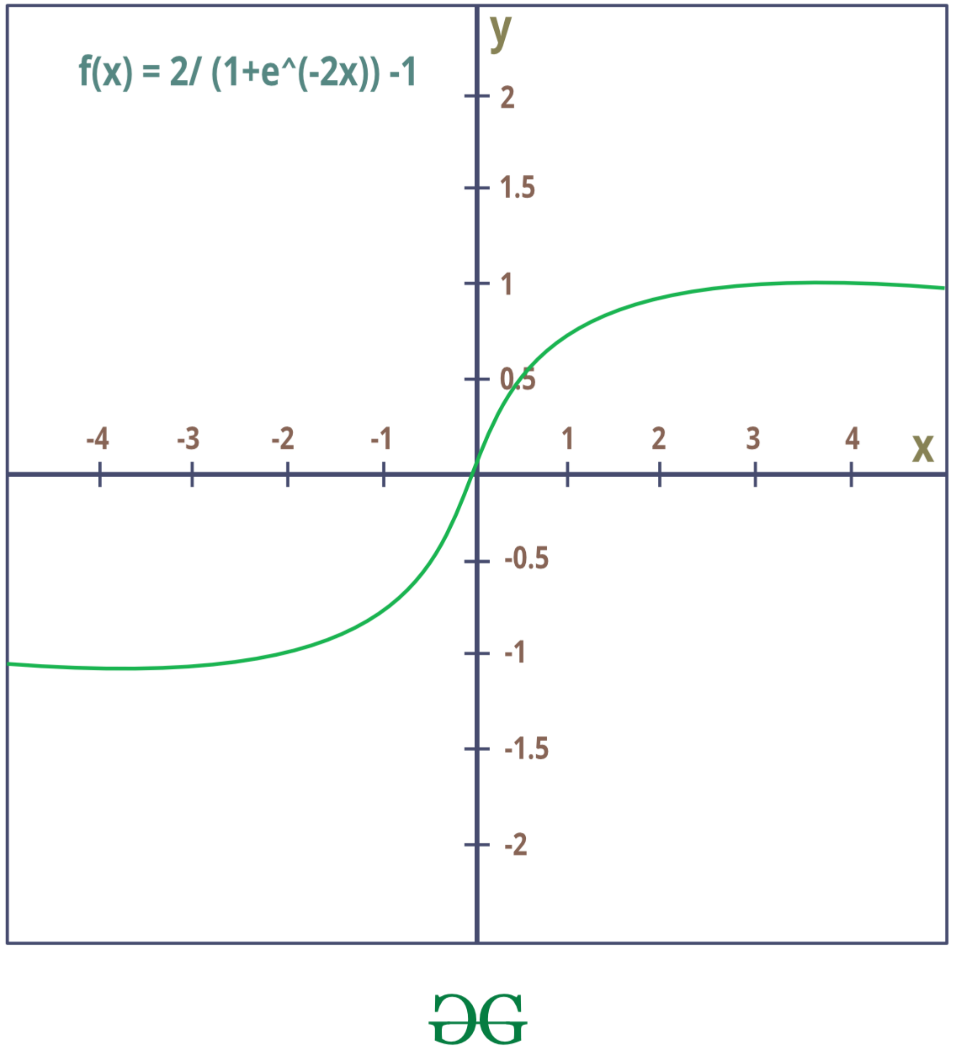
Sigmoid function



Tan h function

- The activation that works almost always better than sigmoid function is Tanh function also known as **Tangent Hyperbolic function**. It’s actually mathematically shifted version of the sigmoid function. Both are similar and can be derived from each other.
- **Equation :-**
  f(x) = tanh(x) = 2/(1 + e-2x) – 1
  OR
  tanh(x) = 2 * sigmoid(2x) – 1
- **Value Range :-** -1 to +1
- **Nature :** non-linear
- **Uses :-** Usually used in hidden layers of a neural network as it’s values lies between **-1 to 1** hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in *centering the data* by bringing mean close to 0. This makes learning for the next layer much easier.

Tanh



RELU Function

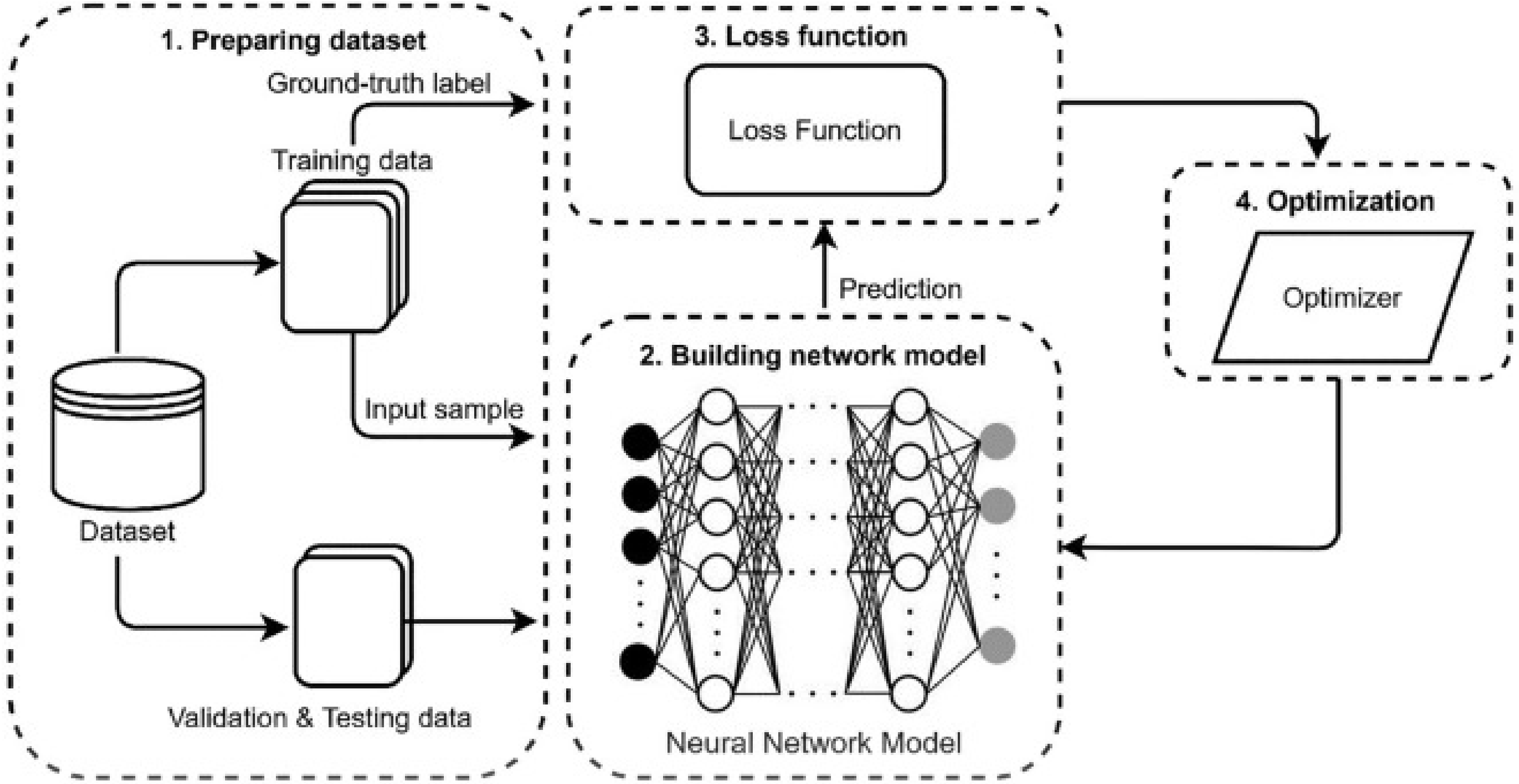
- It Stands for *Rectified linear unit*. It is the most widely used activation function. Chiefly implemented in *hidden layers* of Neural network.
- **Equation :- *A(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise.
- **Value Range :-** [0, inf)
- **Nature :-** non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.
- **Uses :-** ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.

softmax function

- The softmax function is also a type of sigmoid function but is handy when we are trying to handle multi- class classification problems.
- **Nature :-** non-linear
- **Uses :-** Usually used when trying to handle multiple classes. the softmax function was commonly found in the output layer of image classification problems.The softmax function would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs.
- **Output:-** The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.

- The basic rule of thumb is if you really don’t know what activation function to use, then simply use RELU as it is a general activation function in hidden layers and is used in most cases these days.
- If your output is for binary classification then, sigmoid function is very natural choice for output layer.
- If your output is for multi-class classification then, Softmax is very useful to predict the probabilities of each classes.

Training neural network



Preparing dataset

- Training data: A set of examples, which is used for fitting the weights of connections between neurons in neural networks. The training data often contains pairs of samples **(input sample, ground truth label).** The ground truth label is also called the expected output.
- Validation data: A set of examples, which is used to estimate the model fit during tuning the hyperparameters of the network model, such as the number of hidden layers, the number of neurons in each layer, and so on.
- Test data: A set of examples, which is used to estimate the final network model fit on the training data. For most competitions, training data and validation data are published, while the test data is kept as the basis for the final evaluation of the model.

Training

- input layers, hidden layers, and output layers.

Loss function

- Linear regression
- Binary classification problem
- Multiclass classification problem

optimization

- the optimization process is to find a set of parameters, also called a set of weights, to make the loss value of the loss function as small as possible.
- Gradient descent (GD) is the most commonly used optimization algorithm for training neural networks.
- To find a minimum of the loss function with GD, take steps proportional to the opposite direction of the gradient of the loss function at the current point.