Neural Networks

Lecture 6

Neural Network Architectures (2)

Md. Mijanur Rahman, Prof. Dr.

Dept. of Computer Science and Engineering, Jatiya Kabi Kazi Nazrul Islam University, Bangladesh.

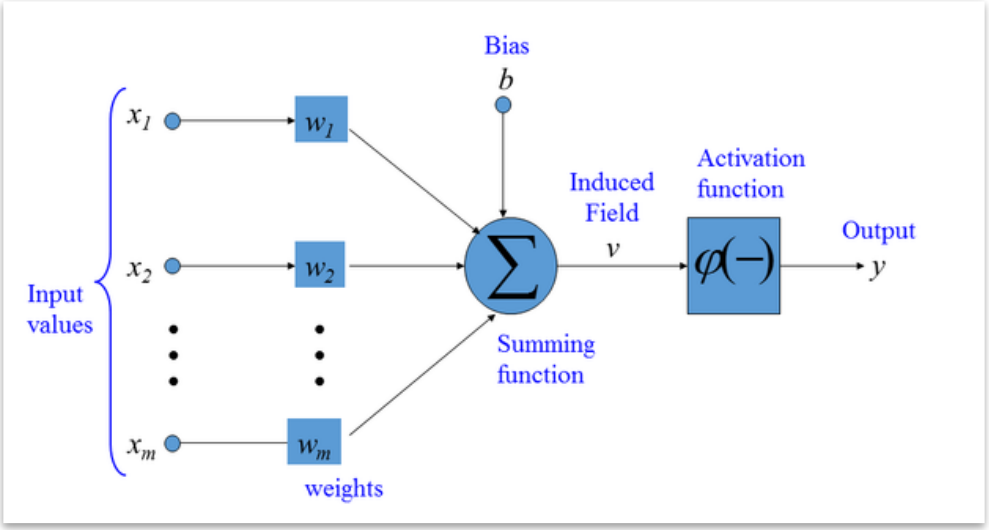
Web: www.mijanrahman.com | Email: mijanjkkniu@gmail.com; mijan@jkkniu.edu.bd

Contents

- **This chapter covers the following topics:**
  - Basic Architecture of an ANN
  - ANN Model and Net Input
  - How do ANNs work?
  - Building Blocks of ANN
  - Network Topology
  - **Learning (or Adjustments of weights)**
  - **Machine Learning Methods**
  - **Activation functions**
  - **McCulloch-Pitts Neuron**

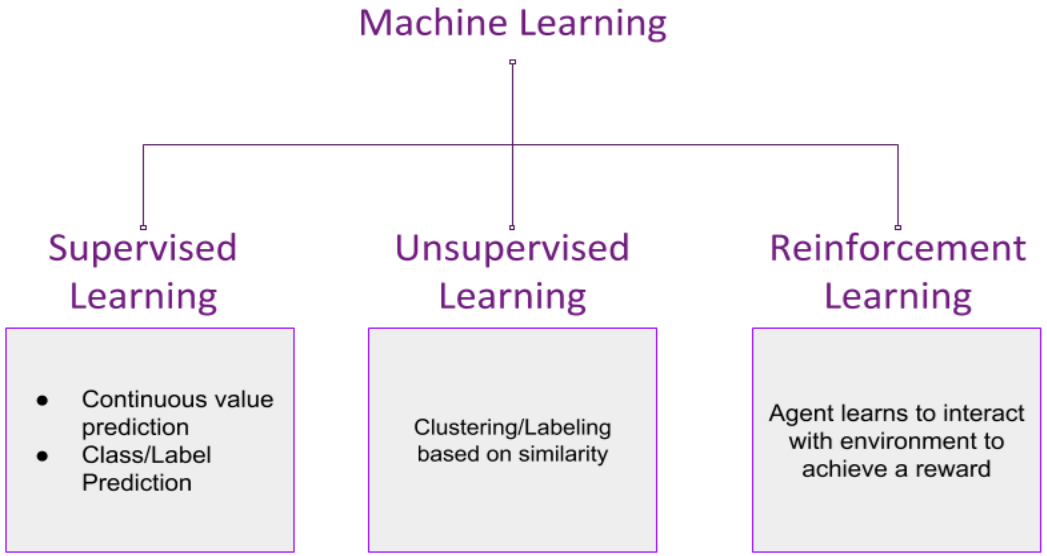
Learning in ANN…

- **Adjustments of Weights**: The method of setting the values of the weights (also known as **training**) is an important distinguishing characteristic of different neural nets. That is, **Learning in ANN** is the technique for changing the weights of associations between the neurons of a specified network.
- Learning in artificial neural networks can be characterized into three different categories, namely **supervised learning**, **unsupervised learning**, and **reinforcement learning**.



Learning in ANN

- Many of the tasks that neural nets can be trained to perform fall into the areas of **mapping, clustering, and constrained optimization**.
- Pattern classification and pattern association may be considered special forms of the more general problem of mapping input vectors or patterns to the specified output vectors or patterns.
- We summarize here the **basic characteristics of supervised, unsupervised, and reinforcement learning**.



Supervised Learning…

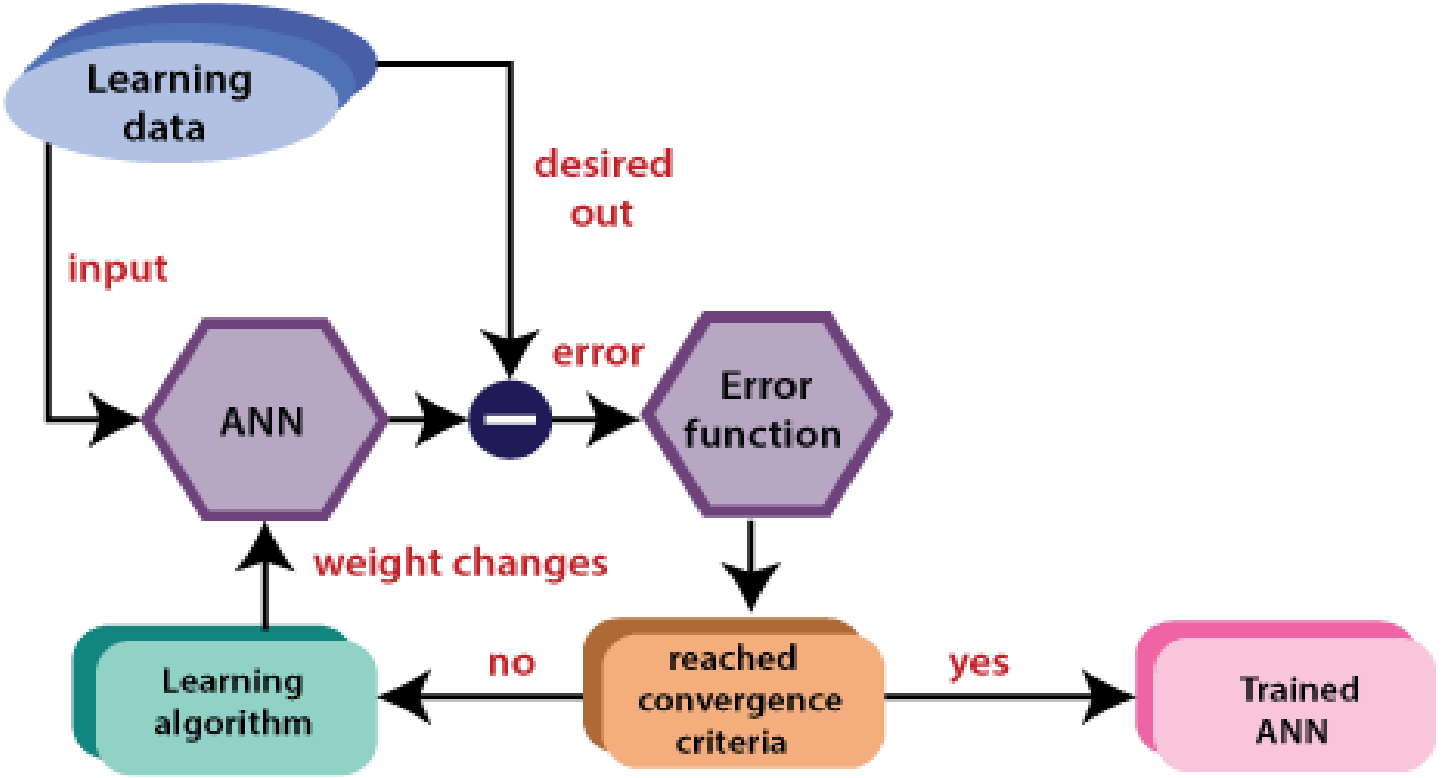
- Supervised learning consists of two words **supervised** and **learning**. Supervise intends to guide. We have supervisors whose duty is to guide and show the way. We can see a similar case in the case of learning.
- Here the machine or program is learning with the help of the existing data set. We have a data set, and we assume the results of new data relying upon the behavior of the existing data sets. It implies the existing data sets acts as a supervisor or boss to find the new data.
- A basic example being electronic gadgets price prediction. The price of electronic gadgets is predicted depending on what is observed with the prices of other digital gadgets.

Supervised Learning…

- The overall learning process is as follows:
  1. During the training of artificial neural networks under supervised learning, the input vector is given to the network, which offers an output vector.
  2. Afterward, the output vector is compared with the desired output vector.
  3. An error signal is produced if there is a difference between the actual output and the desired output vector.
  4. Based on this error signal, the weight is adjusted until the actual output is matched with the desired output.

Supervised Learning

- **The overall learning process:**



Unsupervised Learning…

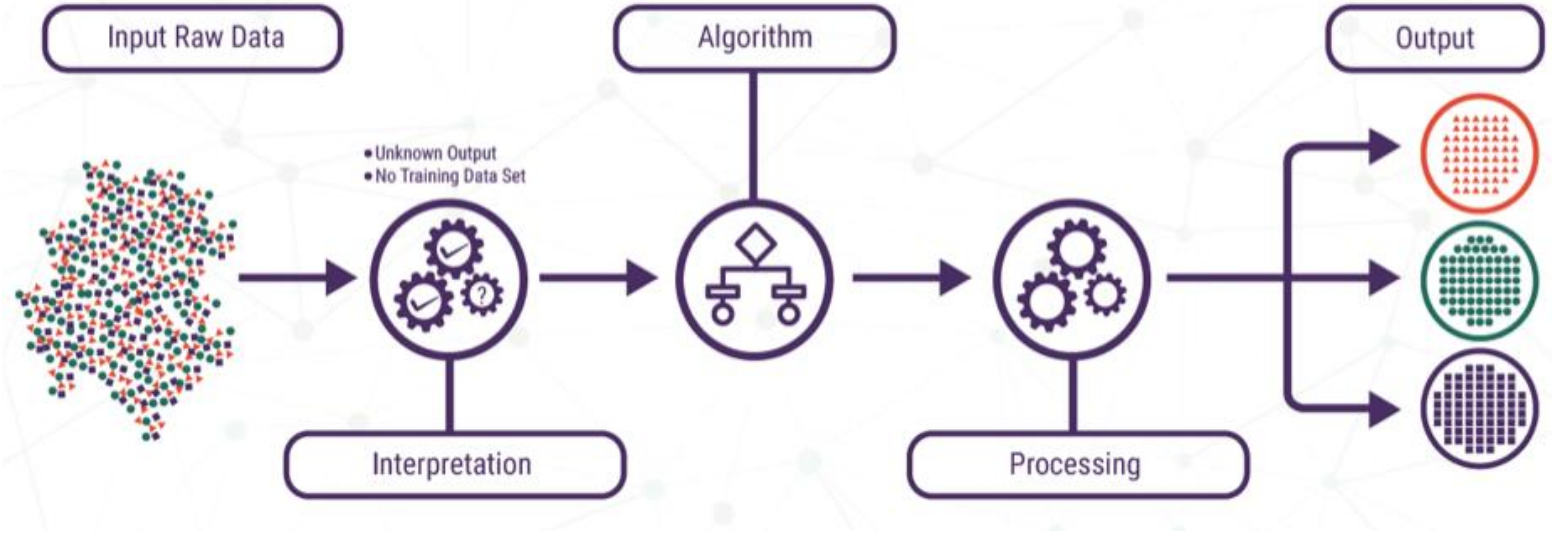
- As the name suggests, unsupervised learning refers to predict something without any supervision or help from existing data.
- In this learning, the net modifies the weights so that the most similar input vectors are assigned to the same output (or cluster) unit. Thus, the data are grouped, relying upon similar characteristics.
- In this situation, there are no existing data to look for direction. A sequence of input vectors is provided, but no target vectors are specified. In other words, there is no supervisor.

Unsupervised Learning…

- **The overall training process is as follows:**
  1. During the training of the artificial neural network under unsupervised learning, the input vectors of a comparative type are joined to form clusters.
  2. At the point when a new input pattern is implemented, then the neural network gives an output response showing the class to which the input pattern belongs.
  3. There is no feedback from the environment about what should be the ideal output and if it is either correct or incorrect.
  4. Consequently, in this type of learning, the network itself must find the patterns and features from the input data and the connection for the input data over the output.

Unsupervised Learning

- **The overall training process:**



Reinforcement Learning…

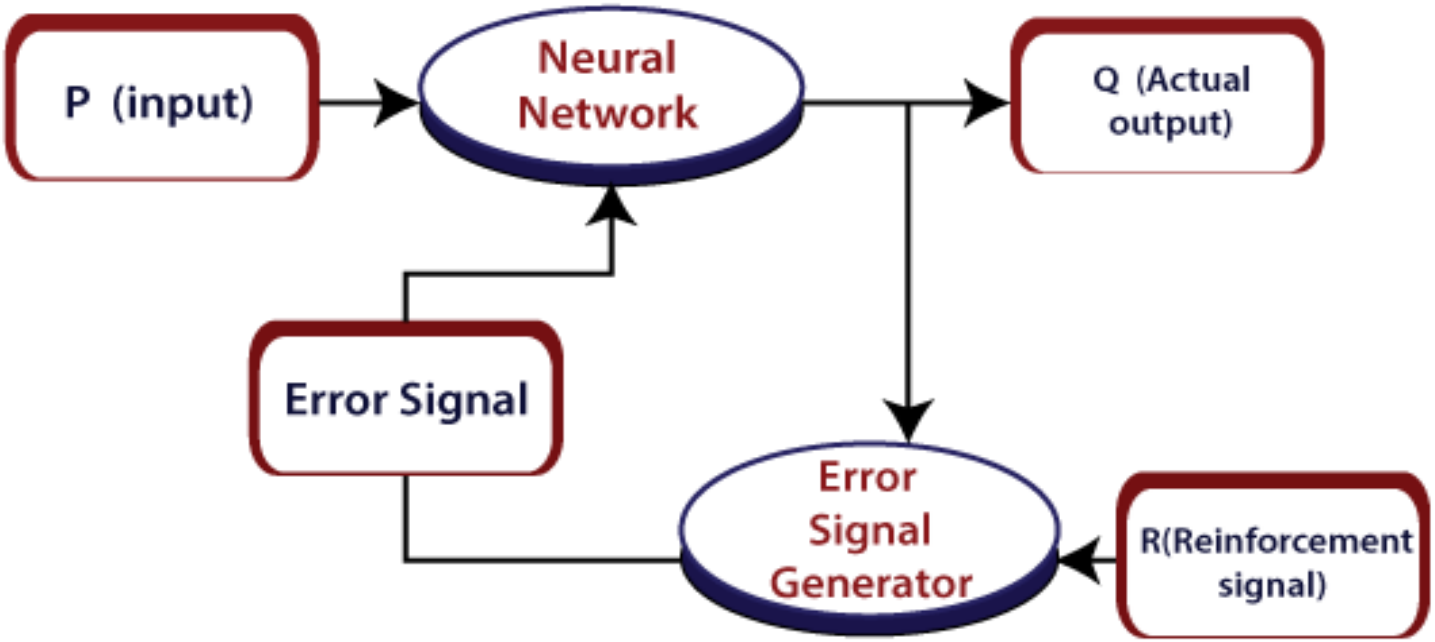
- **Reinforcement learning** (**RL**) is concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward.
- It helps to solve control optimization issues. By using control optimization, we can recognize the best action in each state visited by the system in order to optimize some objective function.
- Reinforcement learning differs from supervised learning in not needing labelled input/output pairs be presented, and in not needing sub-optimal actions to be explicitly corrected. Instead the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

Reinforcement Learning…

- **The basic idea of training is as follows:**
  1. RL involves the training of network model to make a sequence of decisions. In the training, the agent learns to achieve a goal in an uncertain, potentially complex environment. It faces a game-like situation.
  2. The model employs trial and error to come up with a solution to the problem.
  3. The model gets either rewards or penalties for the actions it performs. Its goal is to maximize the total reward.

Reinforcement Learning…

- **The overall training process:**



Activation functions: Definition

- The basic operation of an artificial neuron involves summing its weighted input signal and applying an output, or activation, function.
- **Activation functions** refer to the functions used in neural networks to compute the weighted sum of input and biases, which is used to choose the neuron that can be fire or not.
- It's just a thing function that you use to get the output of node. It is also known as **Transfer Function**.
- It controls the presented information through some gradient processing, normally gradient descent. It produces an output for the neural network that includes the parameters in the data.
- Activation function can either be **linear or non-linear**, relying on the function it shows.

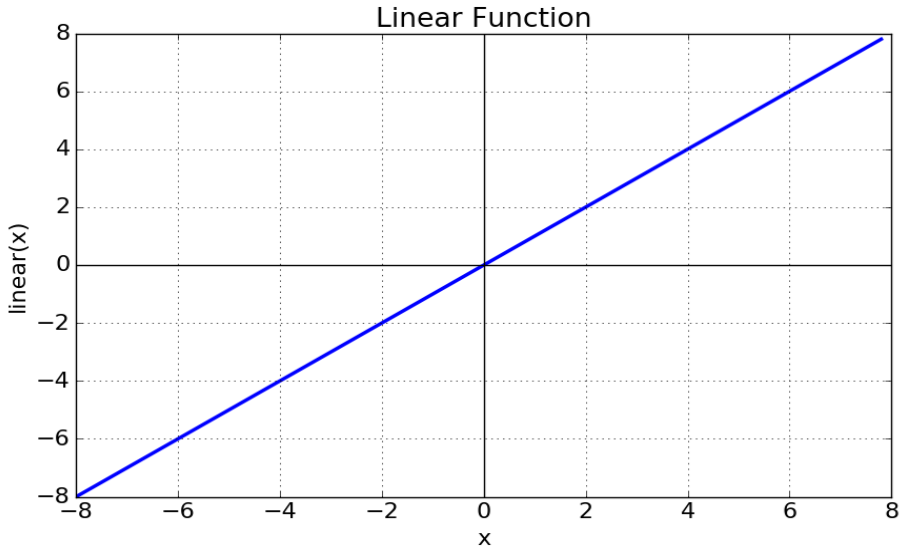
Activation functions: Why

- **Why we use Activation functions with Neural Networks?**
  - *It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).*

- The Activation Functions can be basically divided into 2 types-
  - **Linear Activation Function**
  - **Non-linear Activation Functions**

Activation functions: Types

- **Linear or Identity Activation Function**
  - The equation of the linear activation function is the same as the equation of a straight line i.e., f(x)=x. Therefore, the output of the functions will not be confined between any range. Range : (-infinity to infinity)
  - If we have many layers and all the layers are linear in nature, then the final activation function of the last layer is the same as the linear function of the first layer. Thus, it can be used at only one place that is the output layer.



Activation functions: Types

- **Non-linear Activation Function**
  - The Nonlinear Activation Functions are the most used activation functions. It makes it easy for the model to generalize or adapt with variety of data and to differentiate between the output.
  - The main terminologies needed to understand for nonlinear functions are:
    - *Derivative or Differential: Change in y-axis w.r.t. change in x-axis. It is also known as slope.*
    - *Monotonic function: A function which is either entirely non-increasing or non-decreasing.*

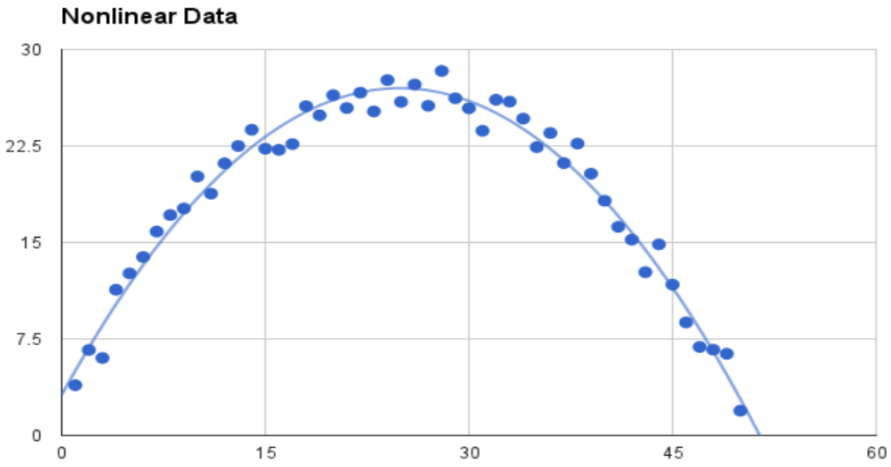


Fig: Non-linear Activation Function

Activation functions: Types

- **Non-linear Activation Function**
- The Nonlinear Activation Functions are mainly divided on the basis of their range or curves-
  1. Sigmoid or Logistic Activation Function
  2. Tanh or hyperbolic tangent Activation Function
  3. ReLU (Rectified Linear Unit) Activation Function
  4. Leaky ReLU

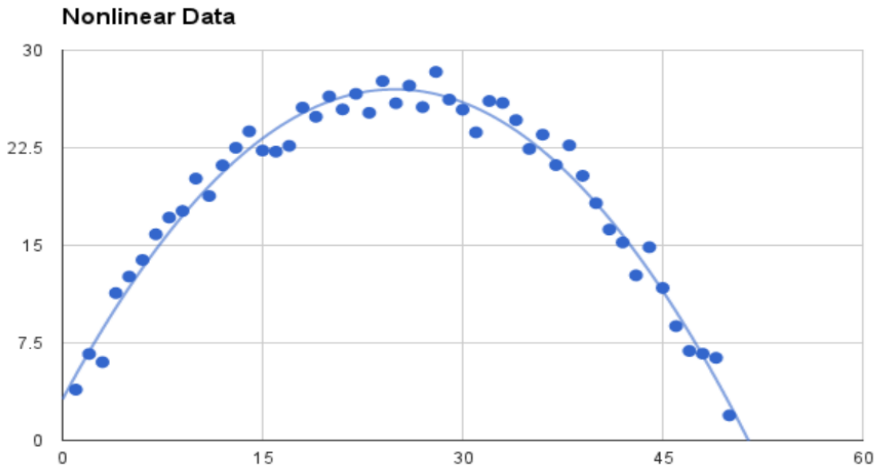
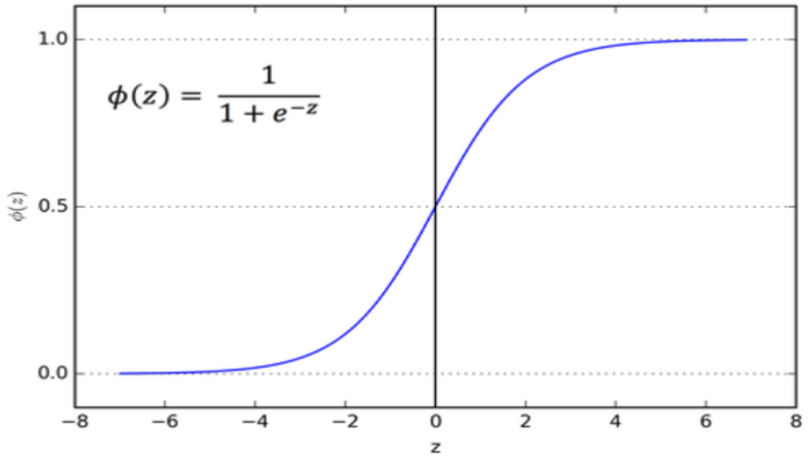


Fig: Non-linear Activation Function

Activation functions: Types

- **Sigmoid or Logistic Activation Function**
  - The Sigmoid Function curve looks like a S-shape. The function is differentiable. That means, we can find the slope of the sigmoid curve at any two points.
  - The main reason of using sigmoid function is, 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.
  - The function is monotonic but function's derivative is not. The logistic sigmoid function can cause a neural network to get stuck at the training time.

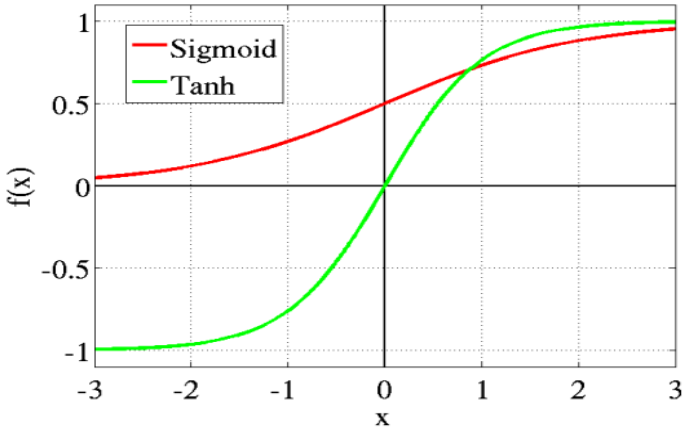
- Fig: The Sigmoid Function Curve



Activation functions: Types

- **Tanh or hyperbolic tangent Activation Function**
  - tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped).
  - 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.
  - The function is differentiable. The function is monotonic while its derivative is not monotonic.
  - The tanh function is mainly used classification between two classes.
  - *Both tanh and logistic sigmoid activation functions are used in feed-forward nets.*

- Fig: The tanh vs. sigmoid Function Curve



Activation functions: Types

- **ReLU (Rectified Linear Unit) Activation Function**
  - The ReLU is 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 shown in Figure, 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.
  - Range: [0 to infinity]. The function and its derivative both are monotonic.
  - 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.

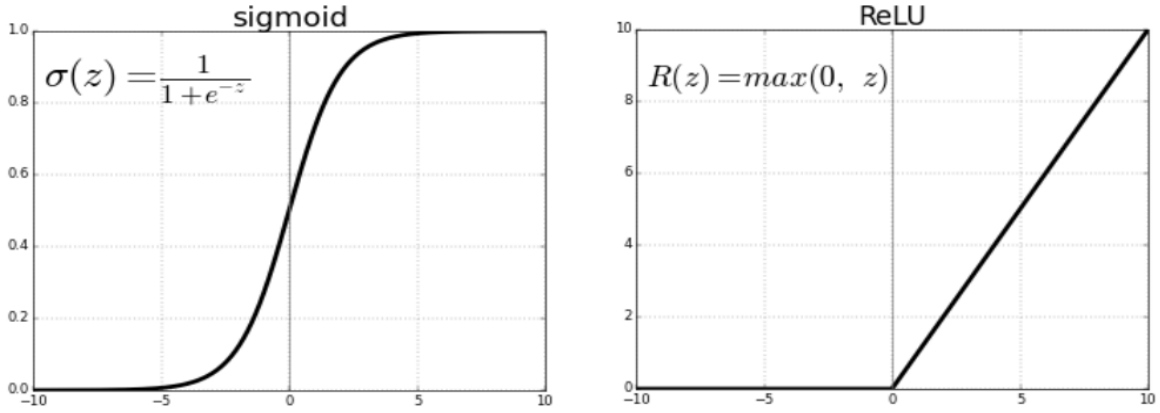


Fig: ReLU v/s Logistic Sigmoid

Activation functions: Types

- **Leaky ReLU Activation Function**
  - 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.
  - Therefore the range of the Leaky ReLU is (-infinity to infinity).
  - Both Leaky and Randomized ReLU functions are monotonic in nature. Also, their derivatives also monotonic in nature.

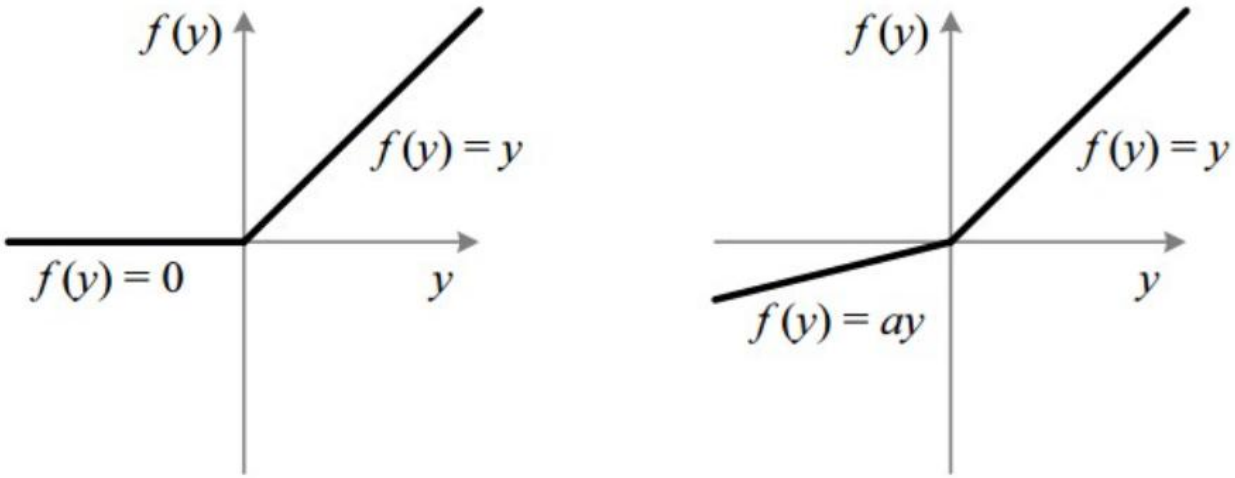
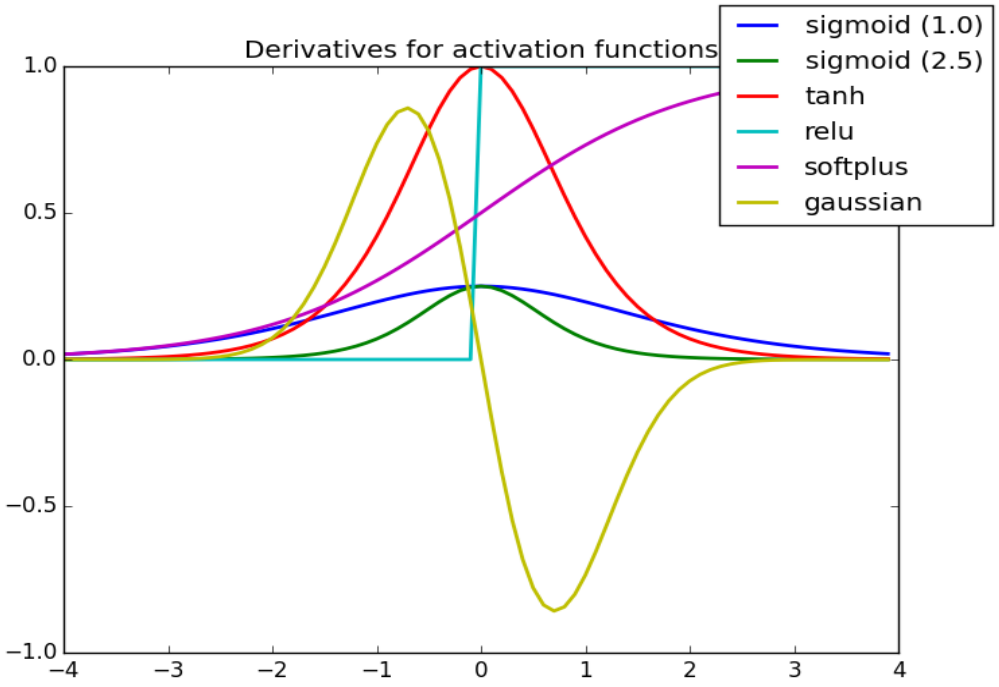


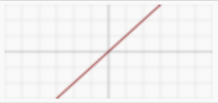
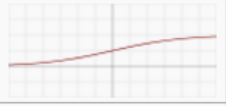
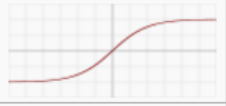
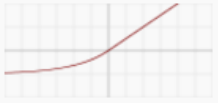
Fig : ReLU v/s Leaky ReLU

Activation functions: Derivative

- **Why derivative/differentiation is used ?**
  - When updating the curve, to know in which direction and how much to change or update the curve depending upon the slope.
  - That is why we use differentiation in almost every part of Machine Learning.



Activation functions: Overview

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

NEURAL NETWORK ARCHITECTURES

To be continued…