Cognixia

A Collabera LEARNING SOLUTIONS COMPANY

Introduction to Artificial Neural Network

Agenda

1. History of Neural Network
2. Introduction to Deep Learning
3. Perceptron
4. Forward Propagation

Introduction

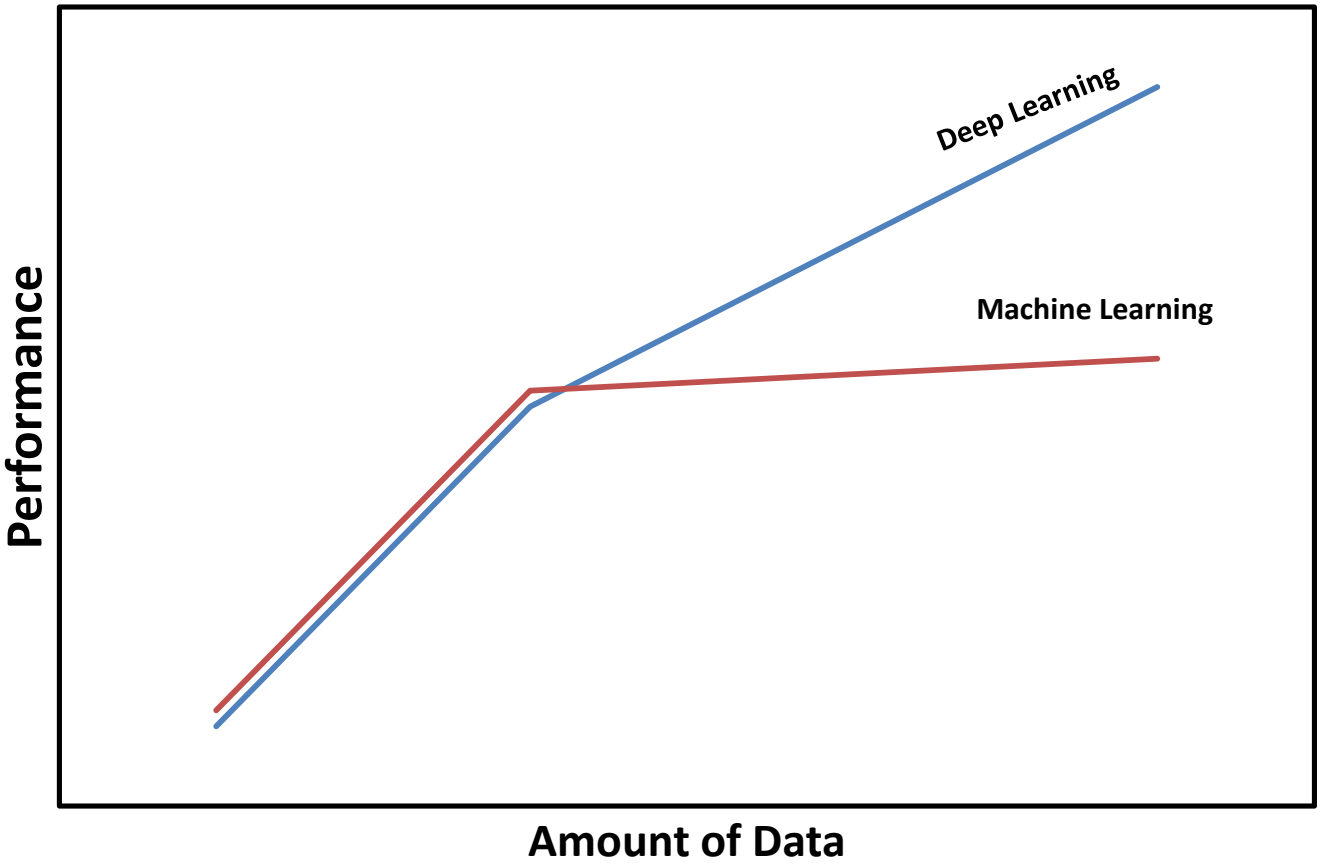
In the evolutionary journey of **artificial intelligence**, the concept of **machine learning** was introduced and in 2015, it became much popular, even more than artificial intelligence itself. Because, empowering machines with learning abilities has been bringing significant changes in the automation industry.

The process of continuous advancement of making machines learn has risen the concept of "**Deep Learning"** . Deep learning explicitly focuses on the development of neural networks.

Neural network enables a computer to classify and organize data similar to the pattern of human brain. Using this approach of categorizing and analyzing the data, a system can make a learned "guess" based on the highest probability. It is even possible to learn from its mistakes, making it "smarter" with time.

So, As Machine learning is said to be part and backbone of Artificial Intelligence. Deep Learning can be termed as part and backbone of machine learning.

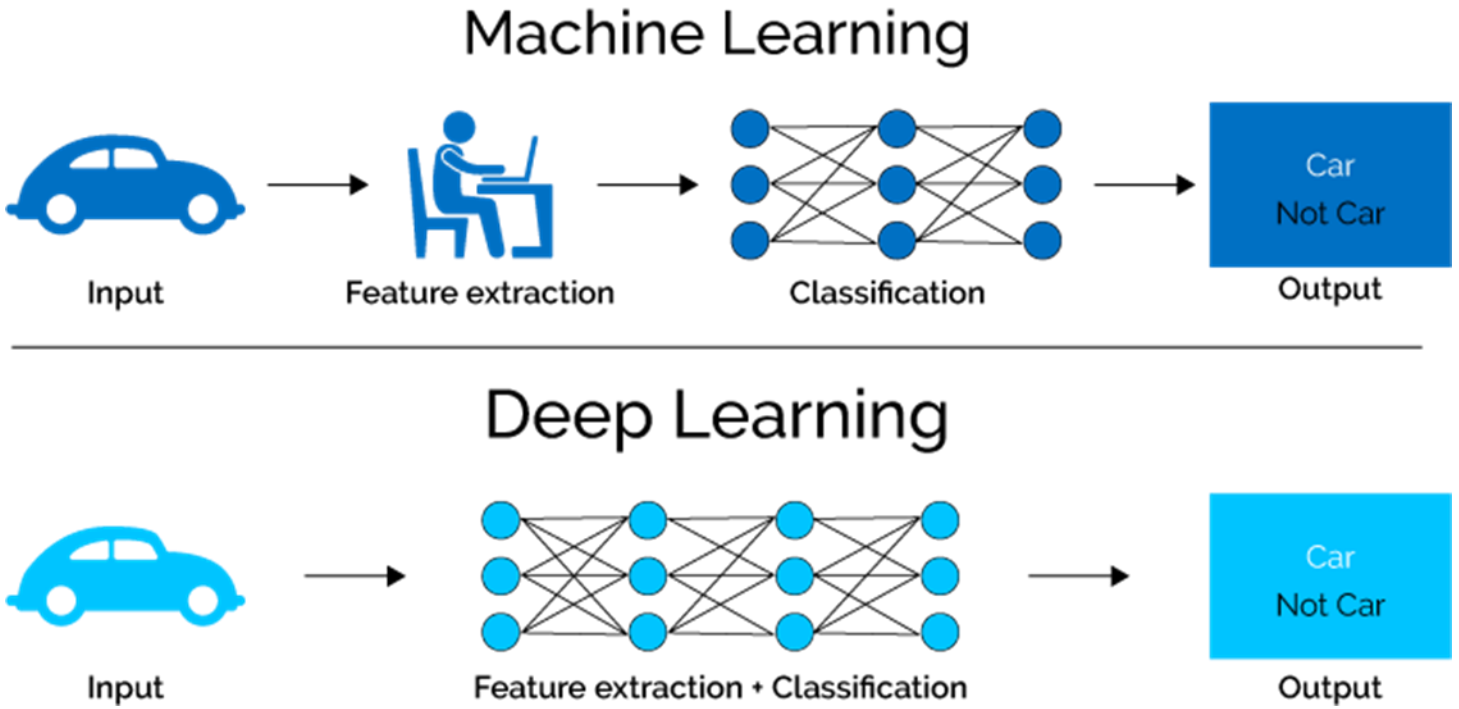
Deep Learning Vs Machine Learning



Core Idea of Deep Learning

Input: The “raw” signal (image, waveform, …)

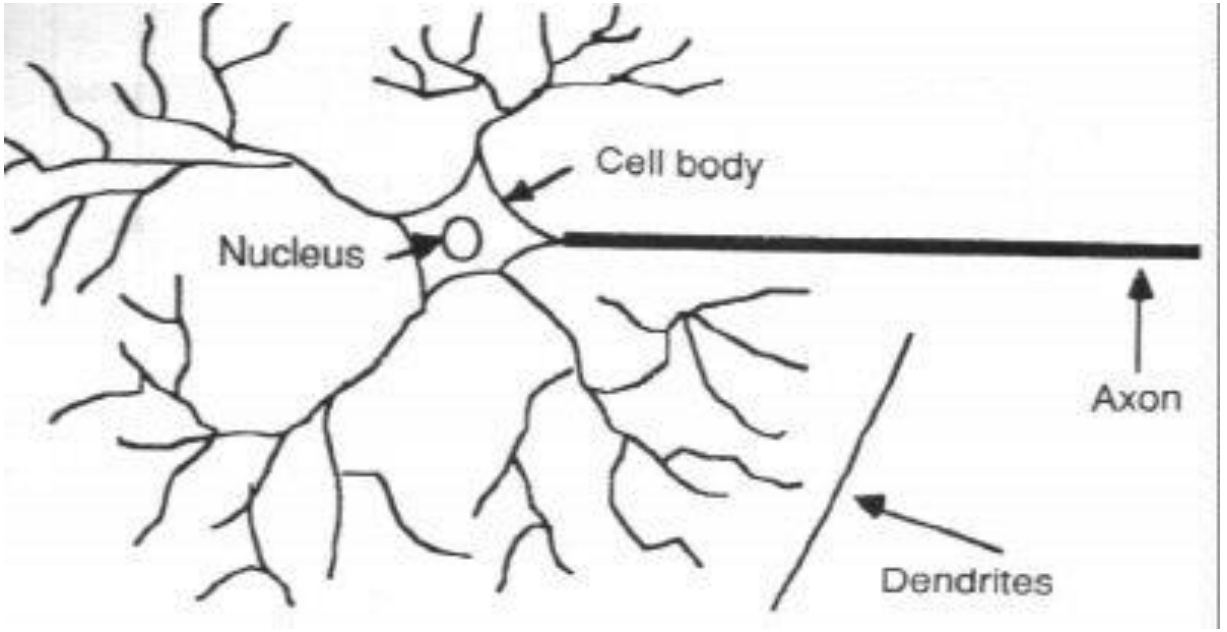
Features: Hierarchy of features is learned from the raw input



Neural Network

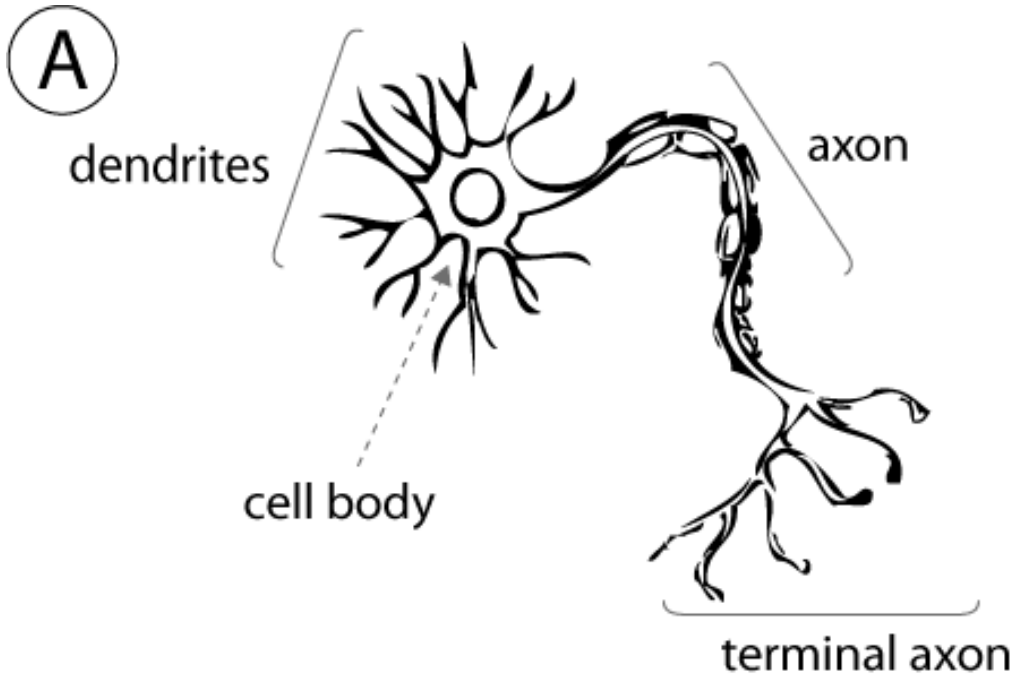
Neural Network models mimic human thought process built by the combination of mathematics and algorithms.

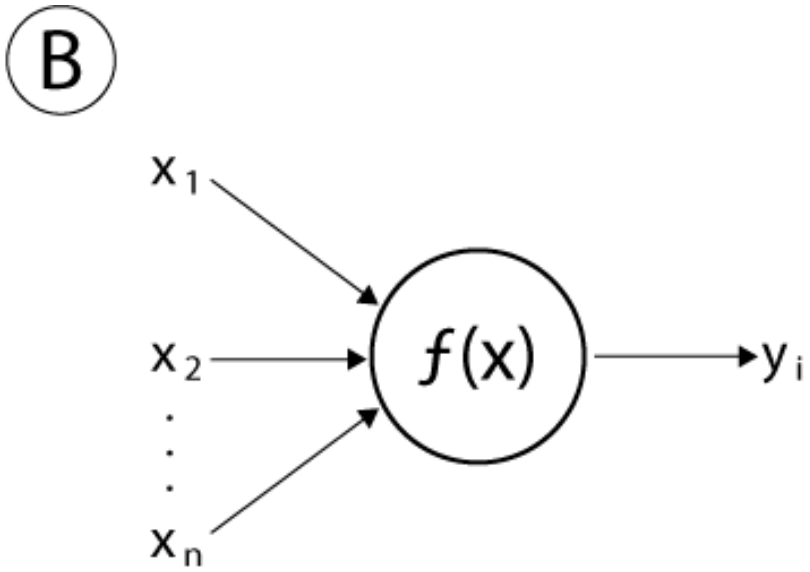
Inside Human Brain



Neural Network

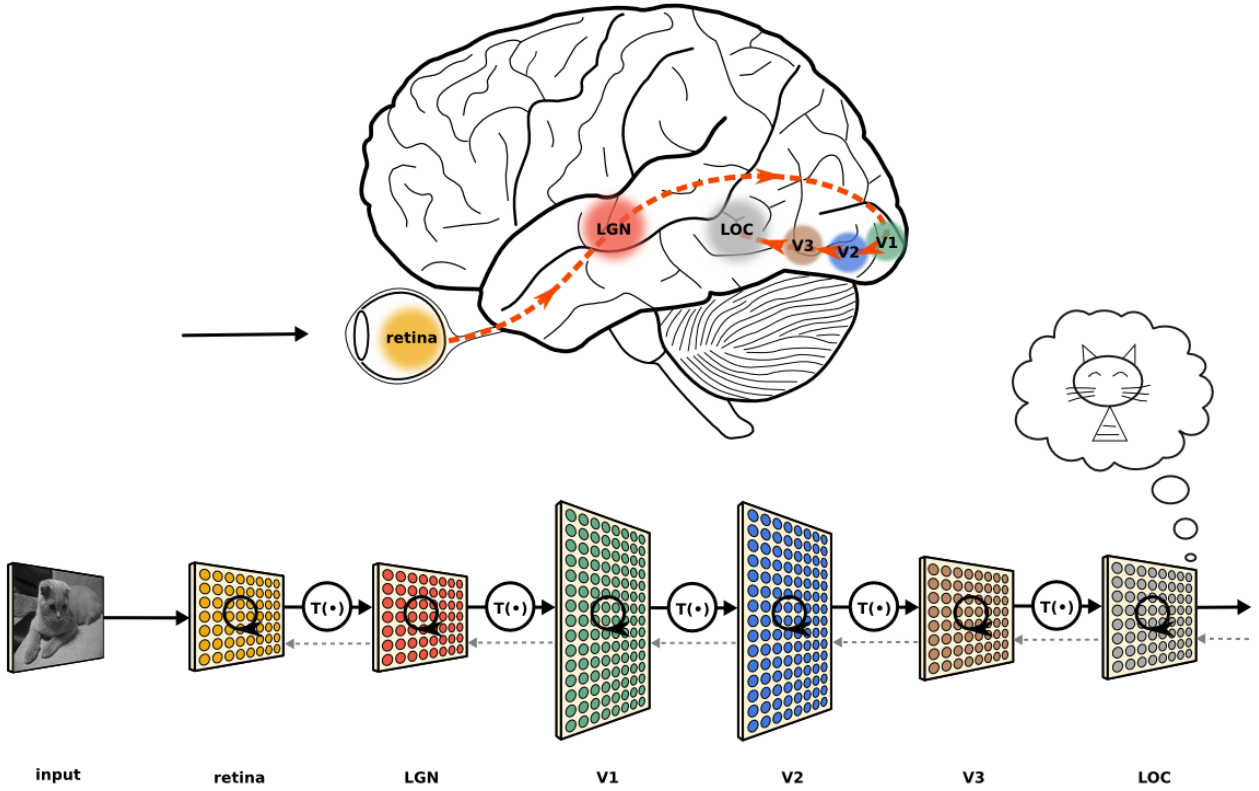
How it is implemented in machine learning:





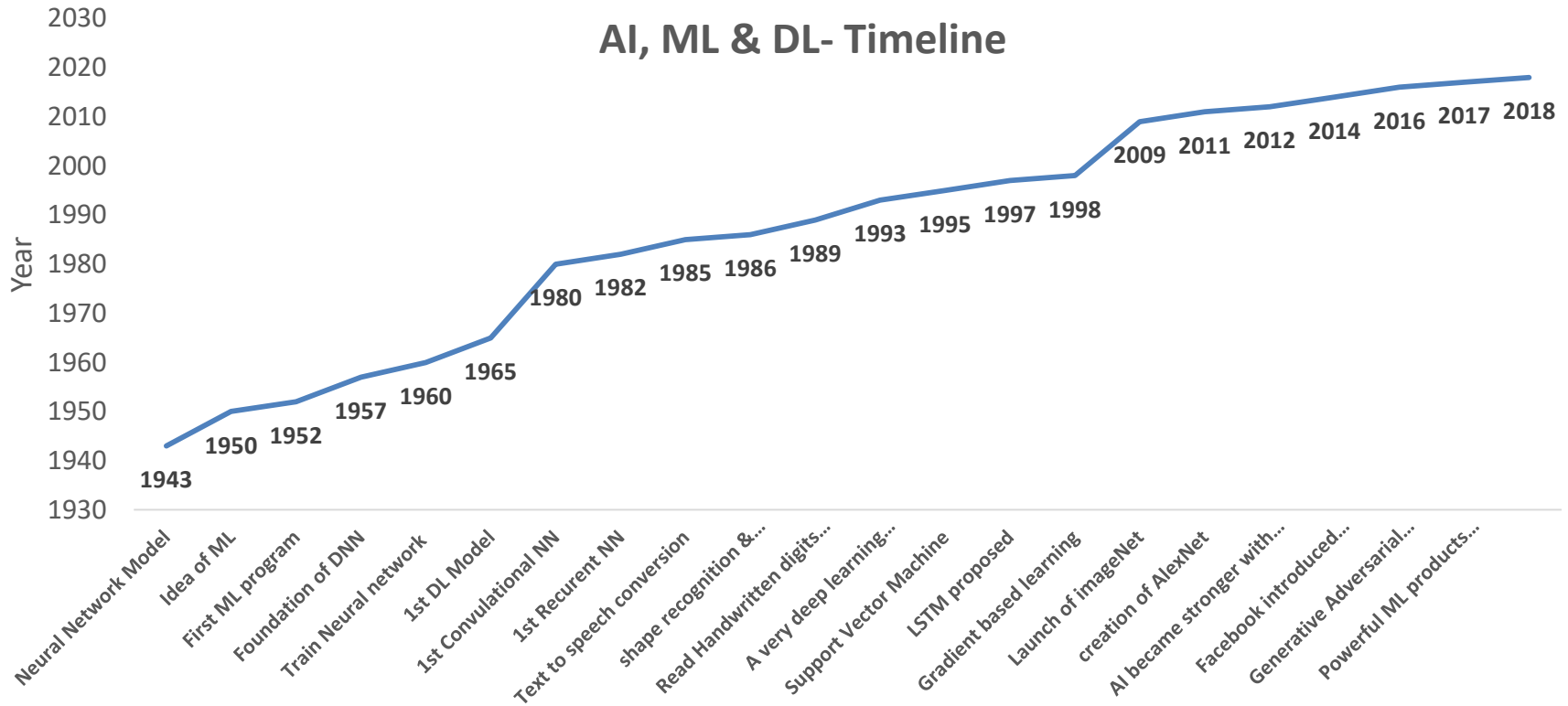
Multilayer Neural Network

To solve complex problems and when data is large enough, single layer neural network is not enough. Then, we have to build multi-layer neural network, just like our human brain does. For Example, to see an object through a human eye, there is a multi layer neural network. Refer the image below:

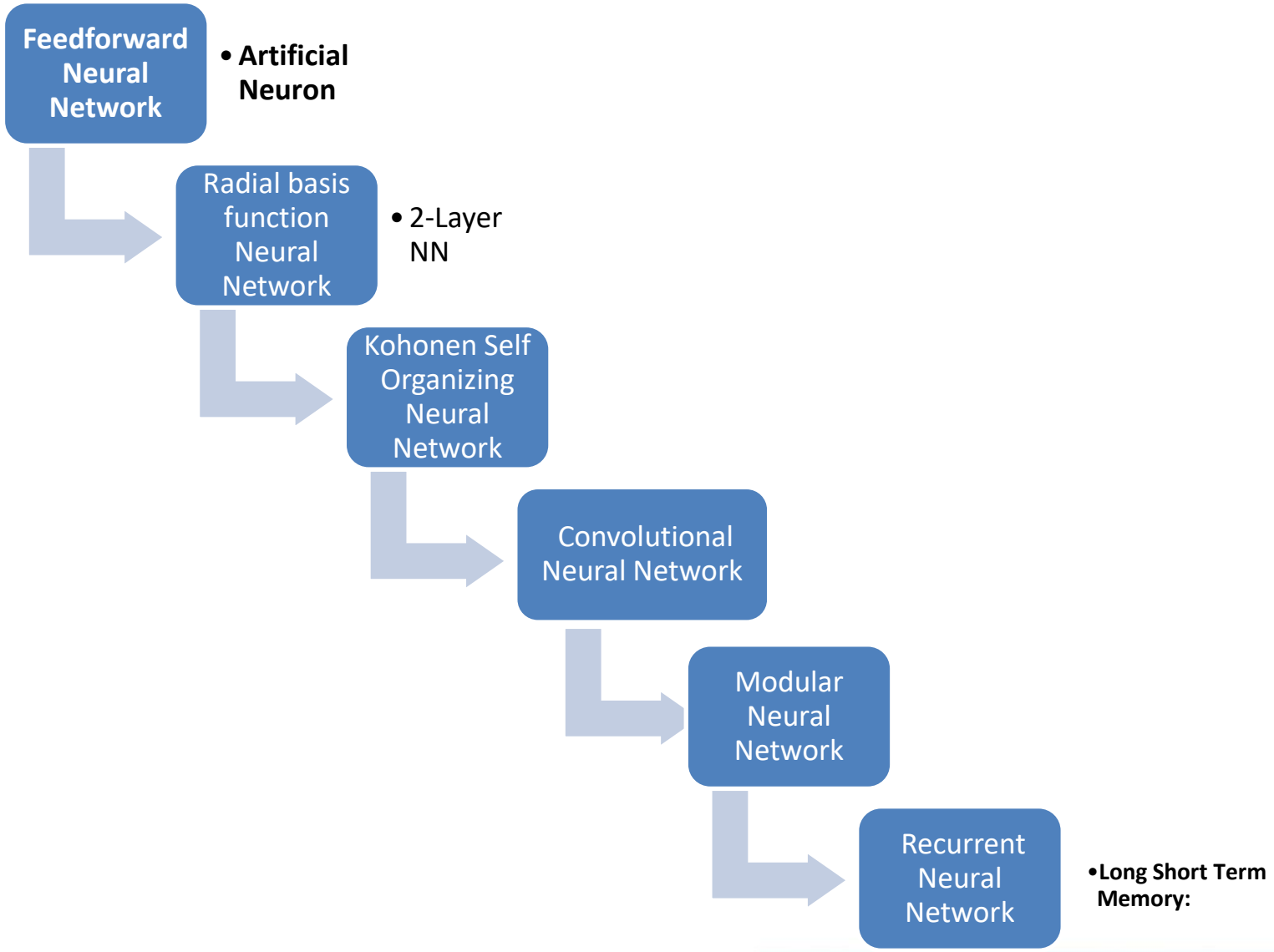


History of Neural Network

Though, studying about deep learning seems to be overwhelming to shape the Artificial intelligence world fantabulously in such short span of time. It's amazing to know that first mathematical model of neural network viz. "McCulloch-Pitts neurons" was actually built in 1943 i.e. before independence!



Types of Neural Network



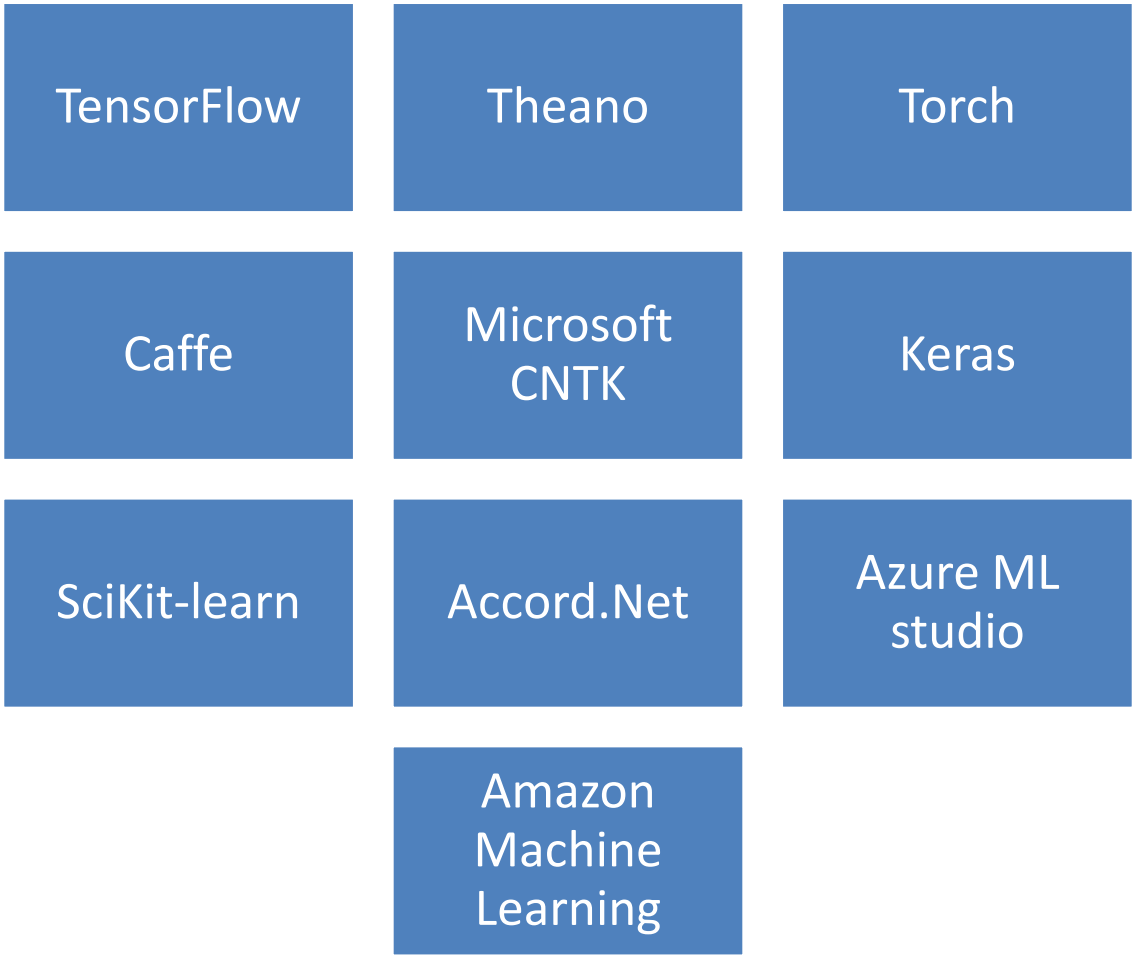
Types of Neural Network: Applications

- ✓ **Kohonen Neural Network** is used to recognize patterns in the data. Its application can be found in medical analysis to cluster data into different categories
- ✓ **ConvNet** are applied in techniques like signal processing and image classification techniques.
- ✓ Application of **Feed forward neural networks** are found in computer vision and speech recognition
- ✓ **Modular neural network** has been applied in Power Restoration Systems
- ✓ The application of **Recurrent Neural Networks** can be found in text to speech (TTS) conversion models

Deep Learning: Applications

- -Speech recognition
- Image classification
- Natural language procession
- Recommendation systems

Libraries available for Deep Learning



Though, we generally use TensorFlow, Keras and Scikit-learn for most of our algorithms

Deep Learning: Keras

- Keras is a high-level neural networks API, written in Python.
- Keras run on top of TensorFlow, Microsoft CNTK and Theano.
- Allows for easy and fast prototyping
- Used for both CNN and RNN, also, for combinations of the two.
- Runs both on CPU and GPU.

Deep Learning: Keras

Keras is built on the following principles:

Modularity: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.

Minimalism: The library provides just enough to achieve an outcome, no frills and maximizing readability.

Extensibility: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.

Python: No separate model files with custom file formats. Everything is native Python.

Keras: Installation

There are two ways to install Keras:

1. Install keras from PYPI
2. Install Keras from Github source

- Install keras from PYPI:
  - sudo pip install keras or,
  - pip install keras

- Install Keras from Github source:
  - **Clone Keras** from Github: git clone https://github.com/keras-team/keras.git
  - **Install**: cd keras sudo python setup.py install

Keras: Available models

- Xception
- VGG16
- VGG19
- ResNet50
- InceptionV3
- InceptionResNetV2
- MobileNet
- DenseNet
- NASNet
- MobileNetV2

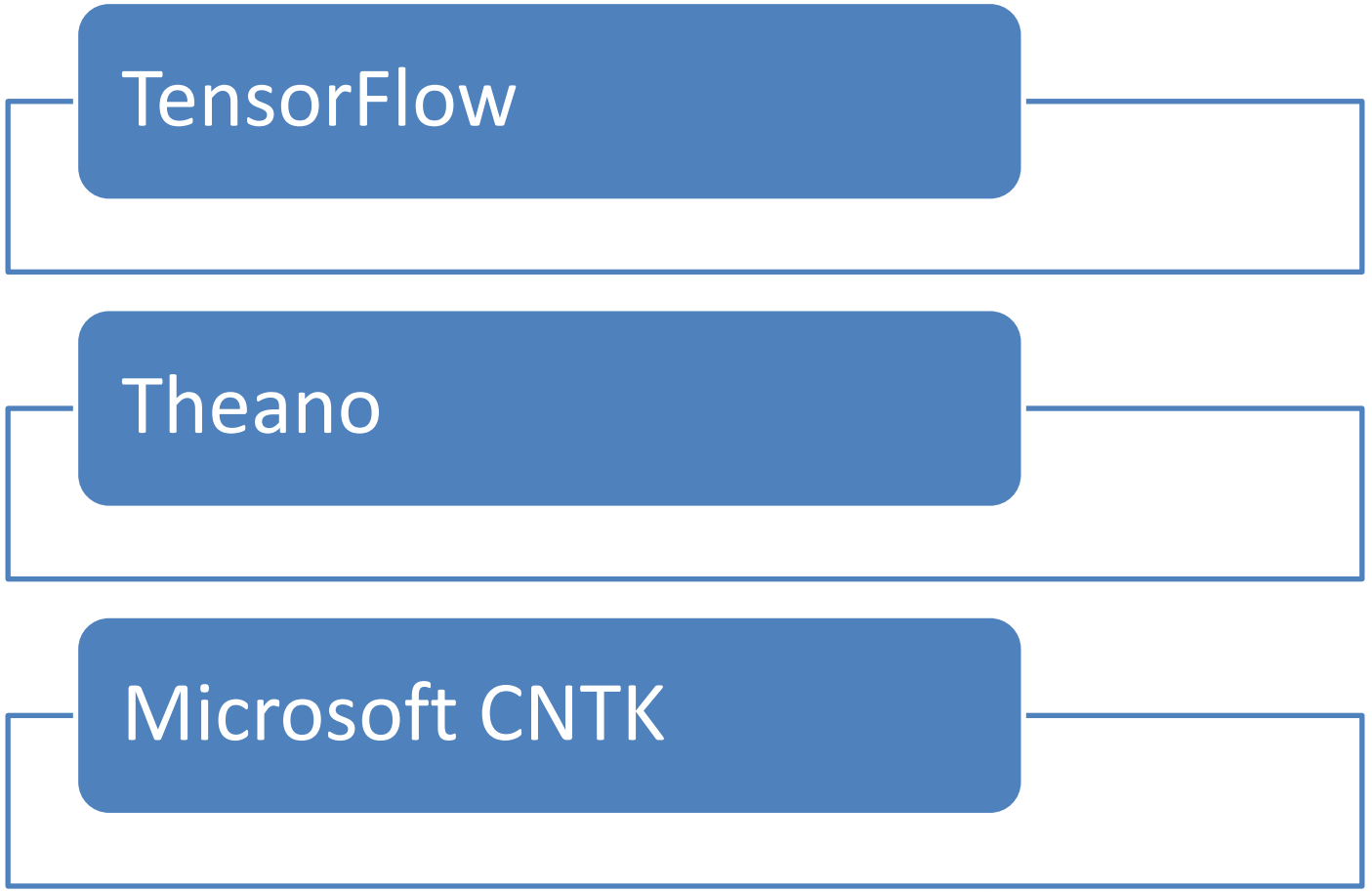
Example: Image Classification Model

```
from keras.applications.resnet50 import ResNet50
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np

model = ResNet50(weights='imagenet')

img_path = 'horse.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch)
print('Predicted:', decode_predictions(preds, top=3)[0])
# Predicted: [(u'n02504013', u'Indian_elephant', 0.82658225), (u'n01871265', u'tusker',
0.1122357), (u'n02504458', u'African_elephant', 0.061040461)]
```

Resnet50: 50 layer Residual Network.

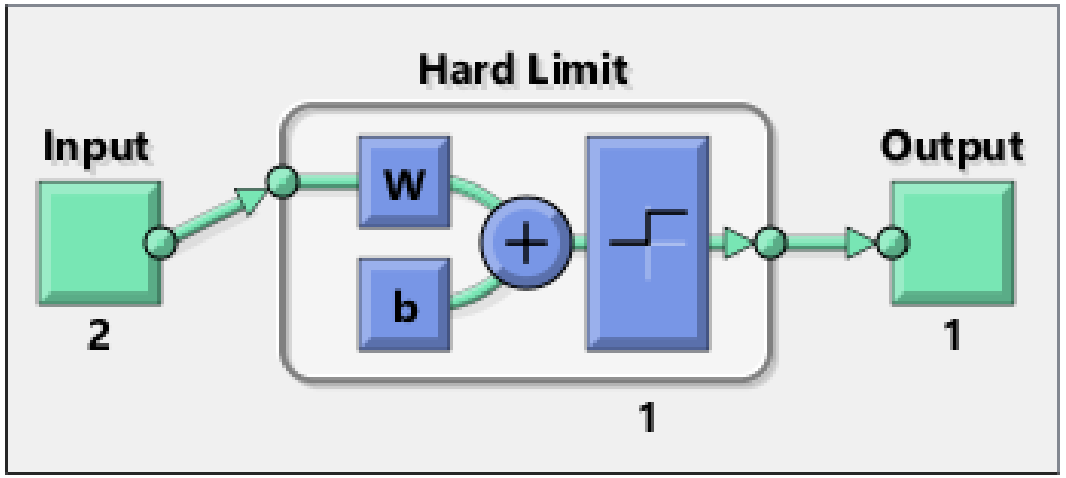
Keras: Backend Implementation

Keras has three backend implementations available



Perceptrons- the First Neural Network

- A Single layer neural network is called as Perceptron.
- It is a linear classifier.
- It is a feed-forward type of neural network



Perceptrons

Perceptron has four parts:

- Input Layers
- Weights and Bias
- Net Sum
- Activation Function

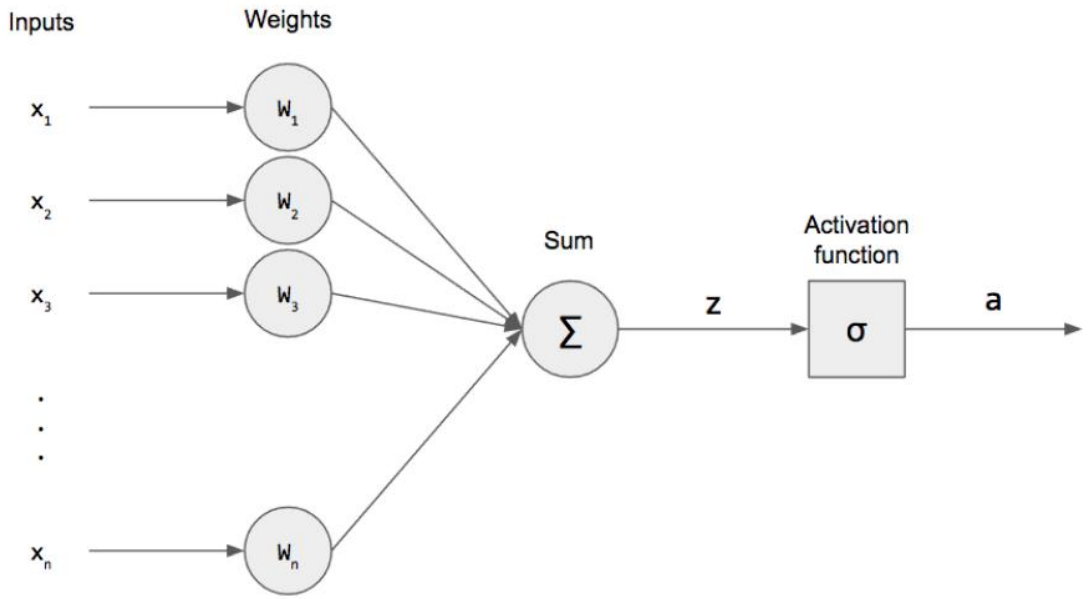
Perceptrons

Activation Function is used to map input values between the required values to make the classification.

- It is also known as ***Transferred function.***

Perceptrons & Neural Networks

Neural Network = Multi-layer Perceptron



Where:
X1, X2,…Xn= Input Vectors

W1, w2…Wn= Weights

X1W1+X2W2+….XnWn= Z (pre-activation)

$$z = \sum_{i=1}^{n} W_i x_i$$

- Perceptrons predict an output based on the current weights and inputs.
- Compares it to the expected output, or label
- Update its weights, if the prediction != the label
- Iterate until the epoch threshold has been reached

Perceptrons & Neural Networks: Example

To update the weights during each iteration, it:

- Finds the error by subtracting the prediction from the label
- Multiplies the error and the learning rate
- Multiplies the result to the inputs
- Adds the resulting vector to the weight vector

Forward Vs Backward Propagation

- To Predict Future Output: **Forward Propagation**

- To Know which factor is responsible for the current output: **Backward Propagation**

Perceptrons & Neural Networks: Example

Example:

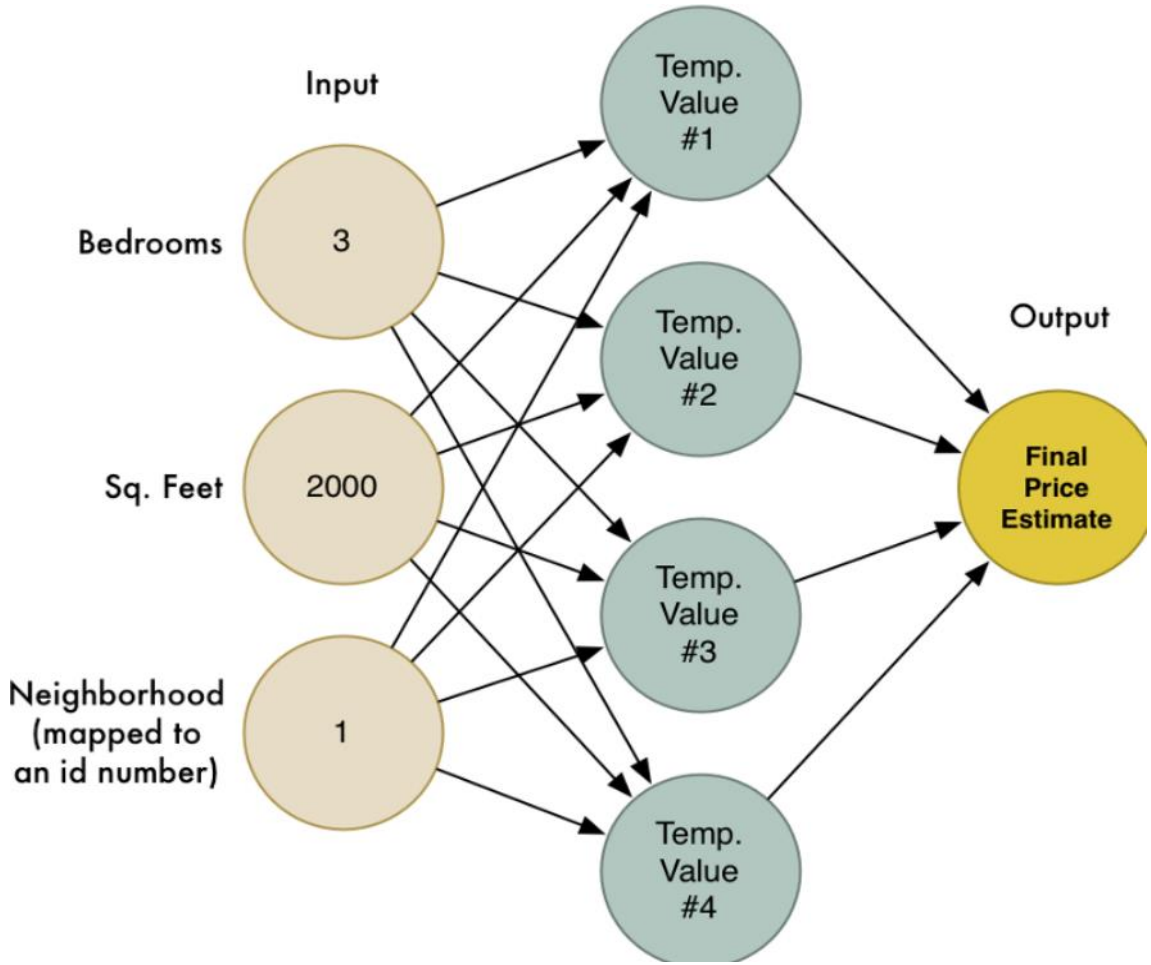
To predict cost value of a house based upon various factors. We can make this estimate by using forward propagation.

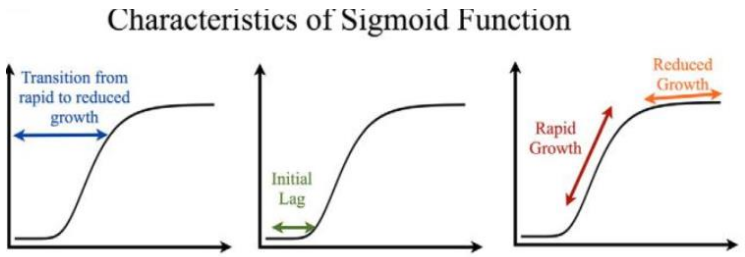
Inputs:

1. No. of bedrooms
2. Area in Sq. Ft
3. Neighbourhood

Refer the bock diagram in next slide

Perceptrons & Neural Networks: Example

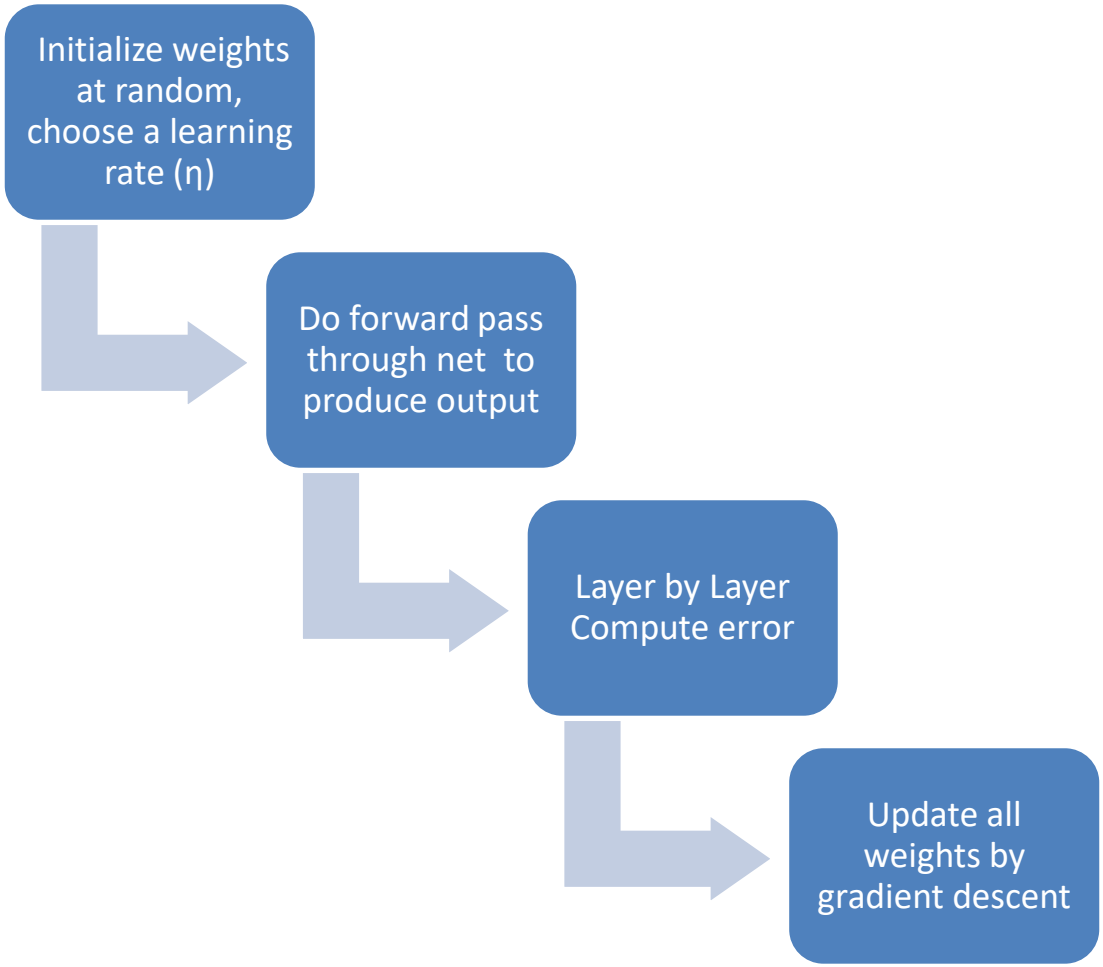




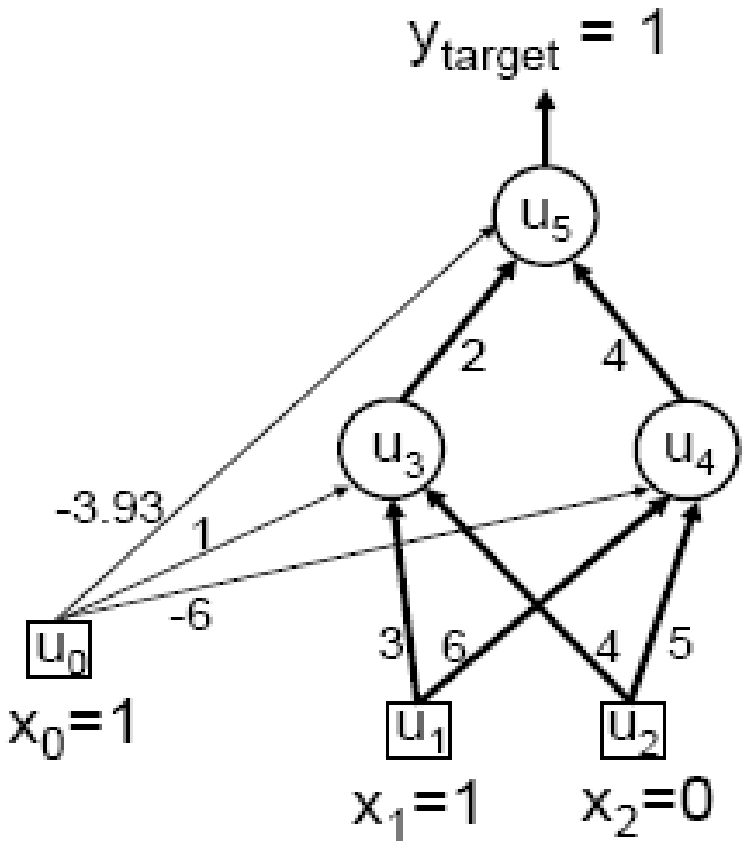
Activation Functions: To determine the output of neural network like yes or no

Sigmoid is a logistic activation function which returns value in either 0 or 1. hence, useful in predicting future output values.

Forward Propagation of Activity



MLP/Backward Propagation : A worked example



Current state:

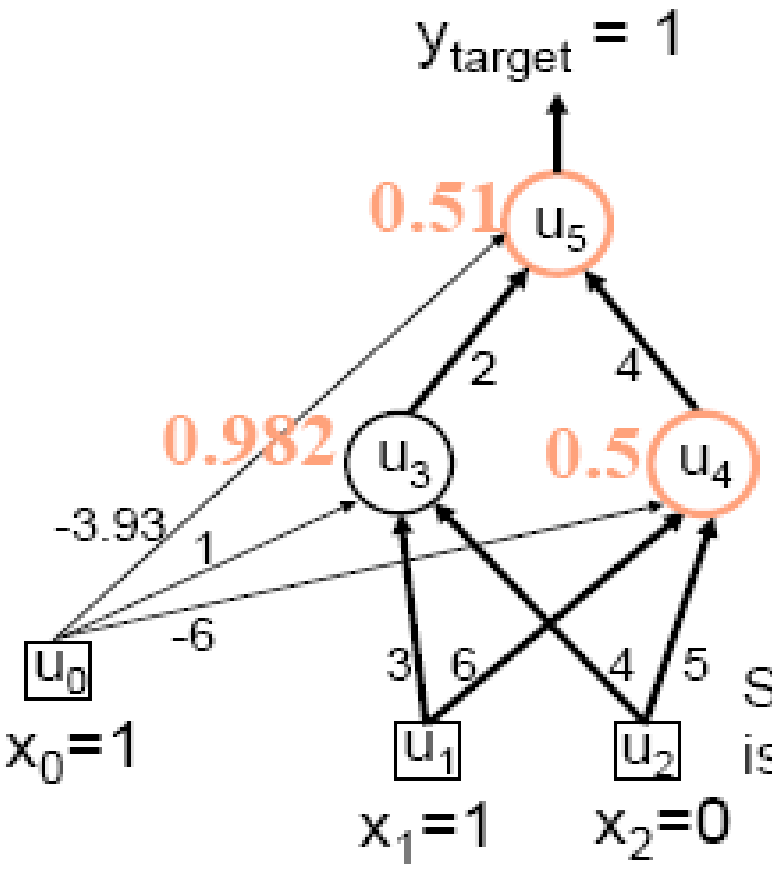
- Weights on arrows e.g. $w_{13} = 3$, $w_{35} = 2$, $w_{24} = 5$
- Bias weights, e.g.

bias for unit 4 (u_4) is $w_{04} = -6$

Training example (e.g. for logical OR problem):

- Input pattern is $x_1=1$, $x_2=0$
- Target output is $y_{target}=1$

Worked example: Forward Pass



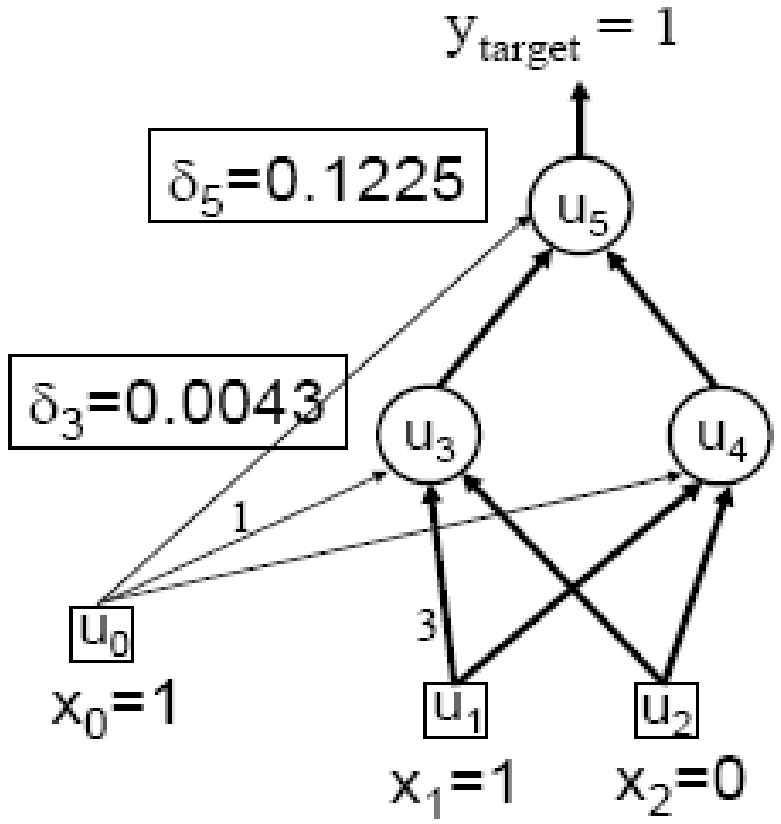
Unit	activation a_j	output y_j
u_3	4.00	0.982
u_4	0.00	0.500
u_5	0.04	**0.510**

(network output)

So the error for this training example is: $(y_{target} - y_5) = (1 - 0.510) = 0.490$

Example: Backward Pass

Update Weights using Generalized Delta Rule (BP)



- Set learning rate η = 0.1

  Change weights by:

  $$\Delta w_{ij} = \eta \delta_j y_i$$

- e.g.bias weight on u_3:

$\Delta w_{03} = \eta \delta_3 x_0$

$=0.1*0.0043*1$

$= 0.0004$

So, new w_{03} ⌫

$w_{03}(old)+\Delta w_{03}$

$=1+0.0004=1.0004$

- and likewise:

  w_{13} ⌫ 3 + 0.0004

  = 3.0004


Cognixia
A Collabera LEARNING SOLUTIONS COMPANY
THANK YOU
www.cognixia.com