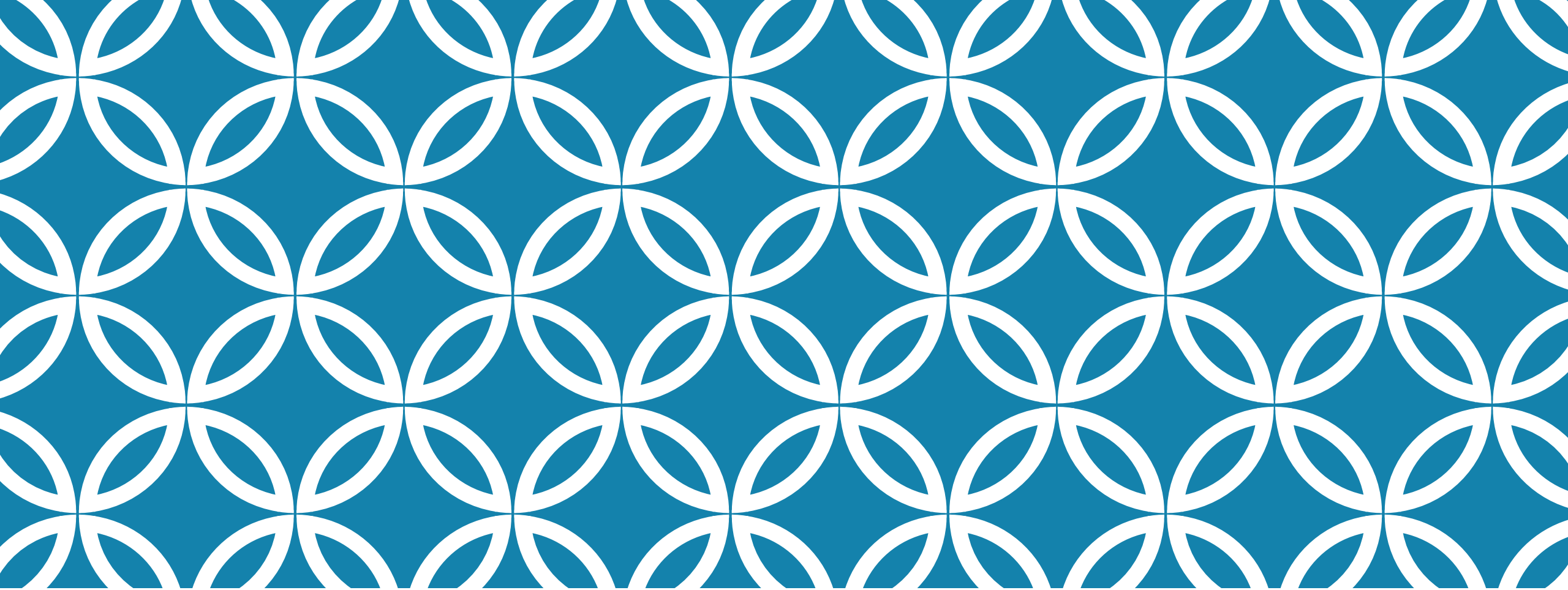


Artificial Neural Networks

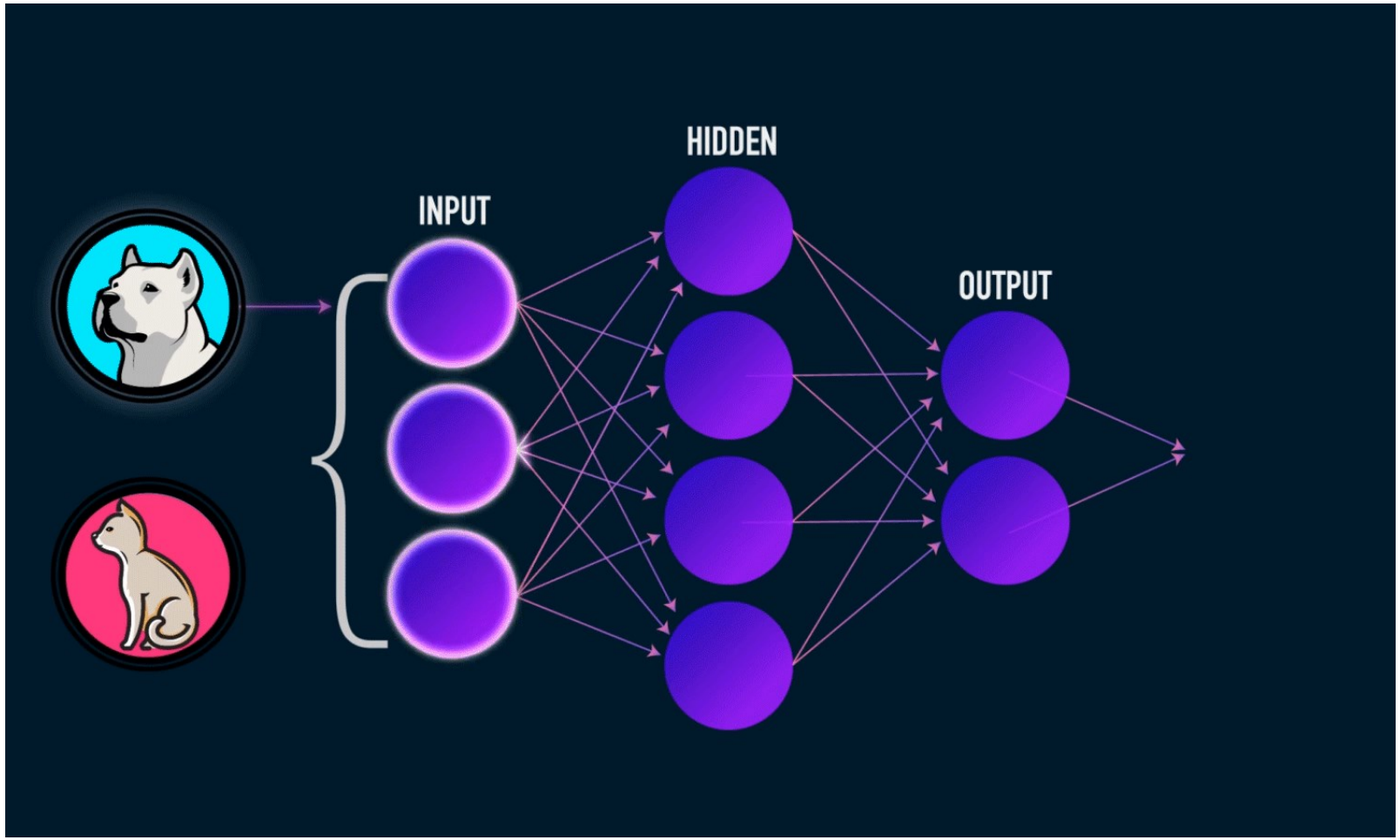
Anshu Pandey

AGENDA

1. Artificial Neural Networks
2. Introduction to Neurons
3. Single Neuron Model
4. Activation Functions
5. Neural Network Architecture
6. Types of Neural Networks
7. Applications, Advantages and Limitations
8. Single Layer Neural Network Code

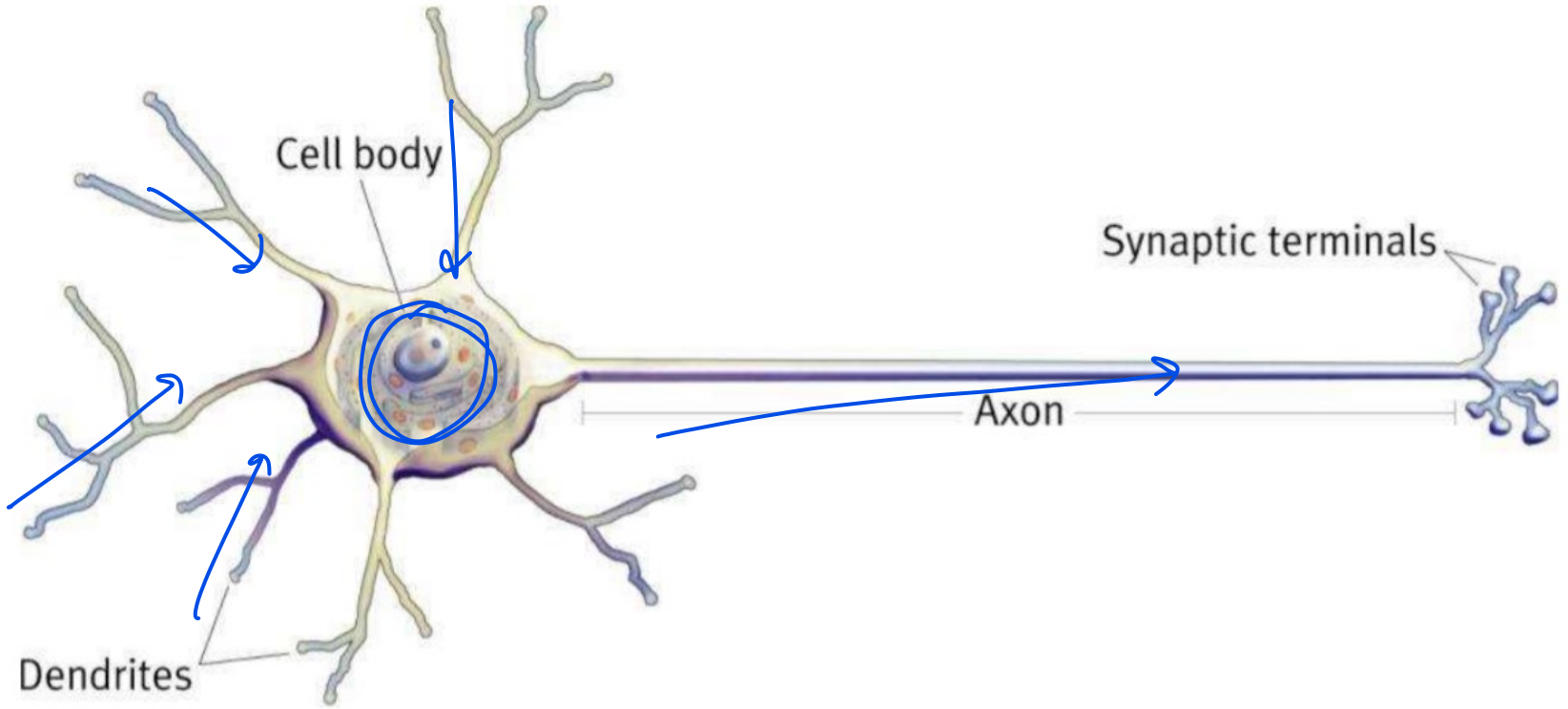
Artificial Neural Networks

Artificial Neural Networks, in general—is a biologically inspired network of artificial neurons configured to perform specific tasks.



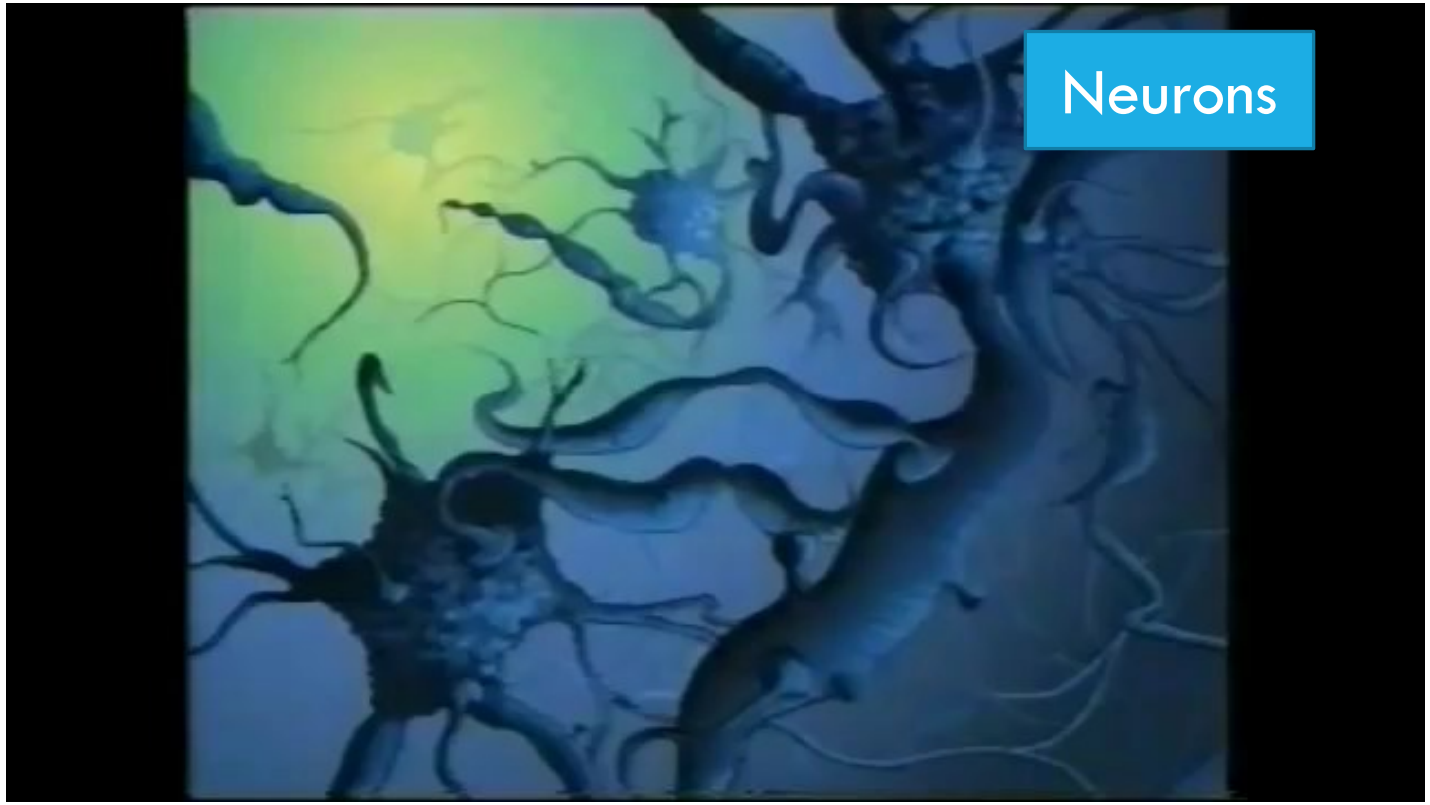
Neural networks learn from examples

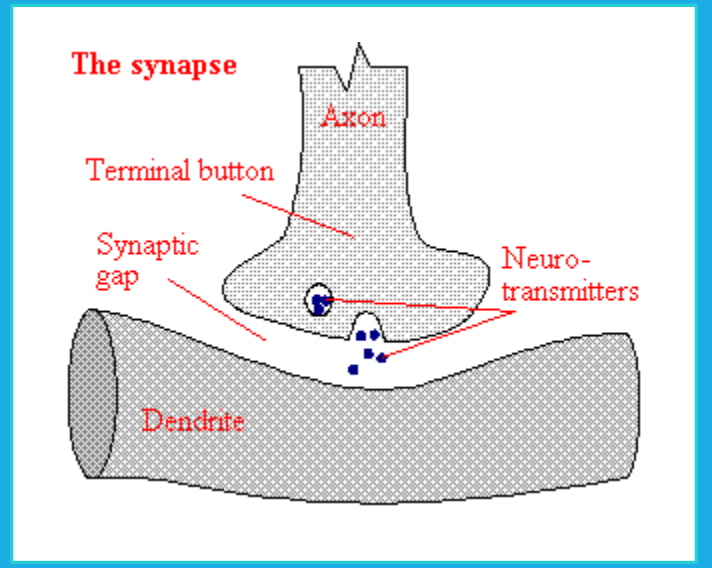
- No requirement of an explicit description of the problem.
- No need for a programmer.
- The neural computer adapts itself during a training period, based on examples of similar problems even without a desired solution to each problem. After sufficient training the neural computer is able to relate the problem data to the solutions, inputs to outputs, and it is then able to offer a viable solution to a brand new problem.
- Able to generalize or to handle incomplete data.



INTRODUCTION TO NEURON

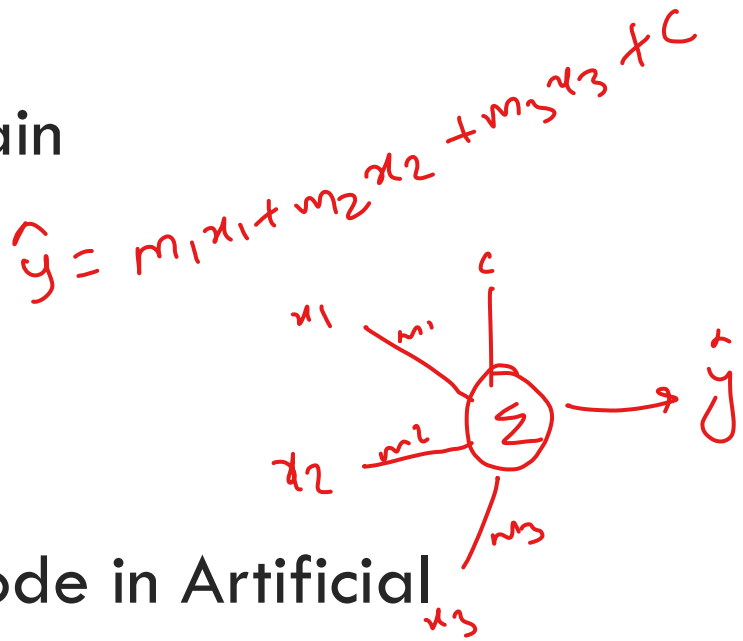
Information exchange b/w Neurons



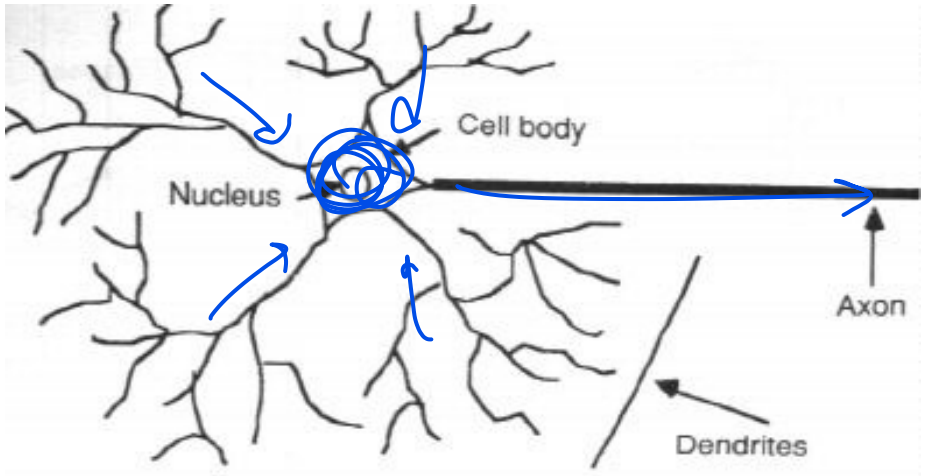


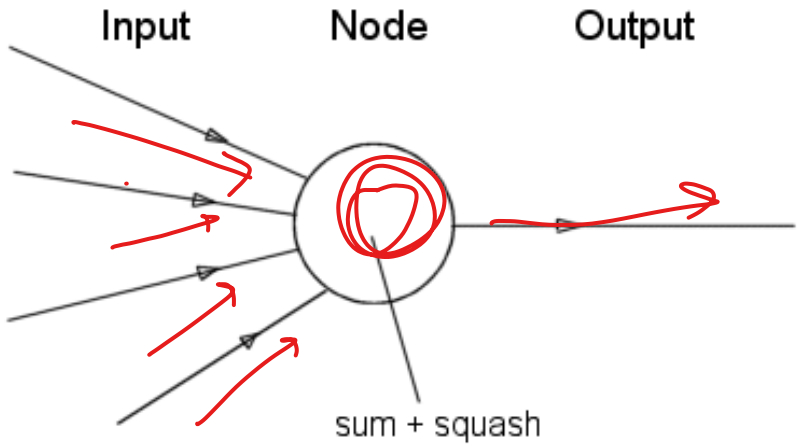
Neuron vs. Node

Neuron in Brain

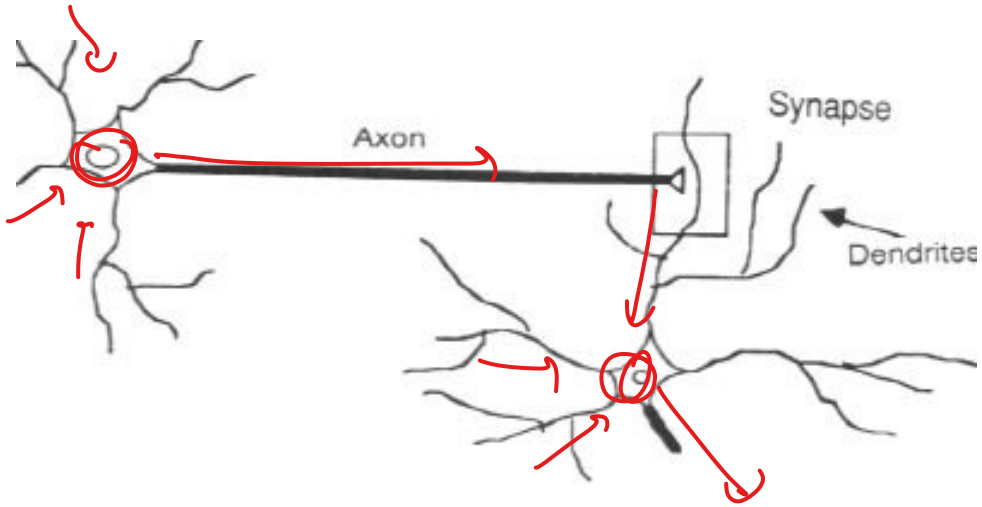


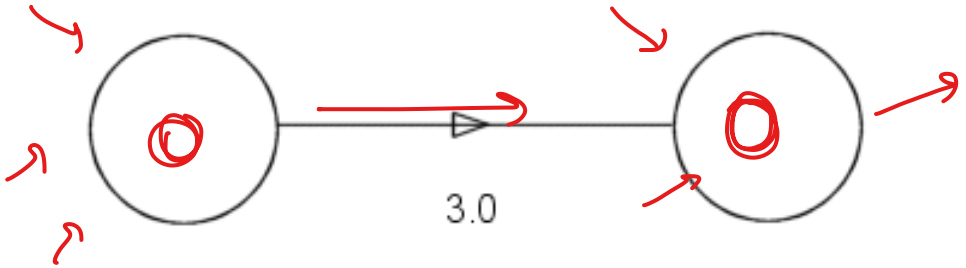
A Neuron/Node in Artificial Neural Network





Synapse vs. weight





Architecture of Neural Network

Dense layers

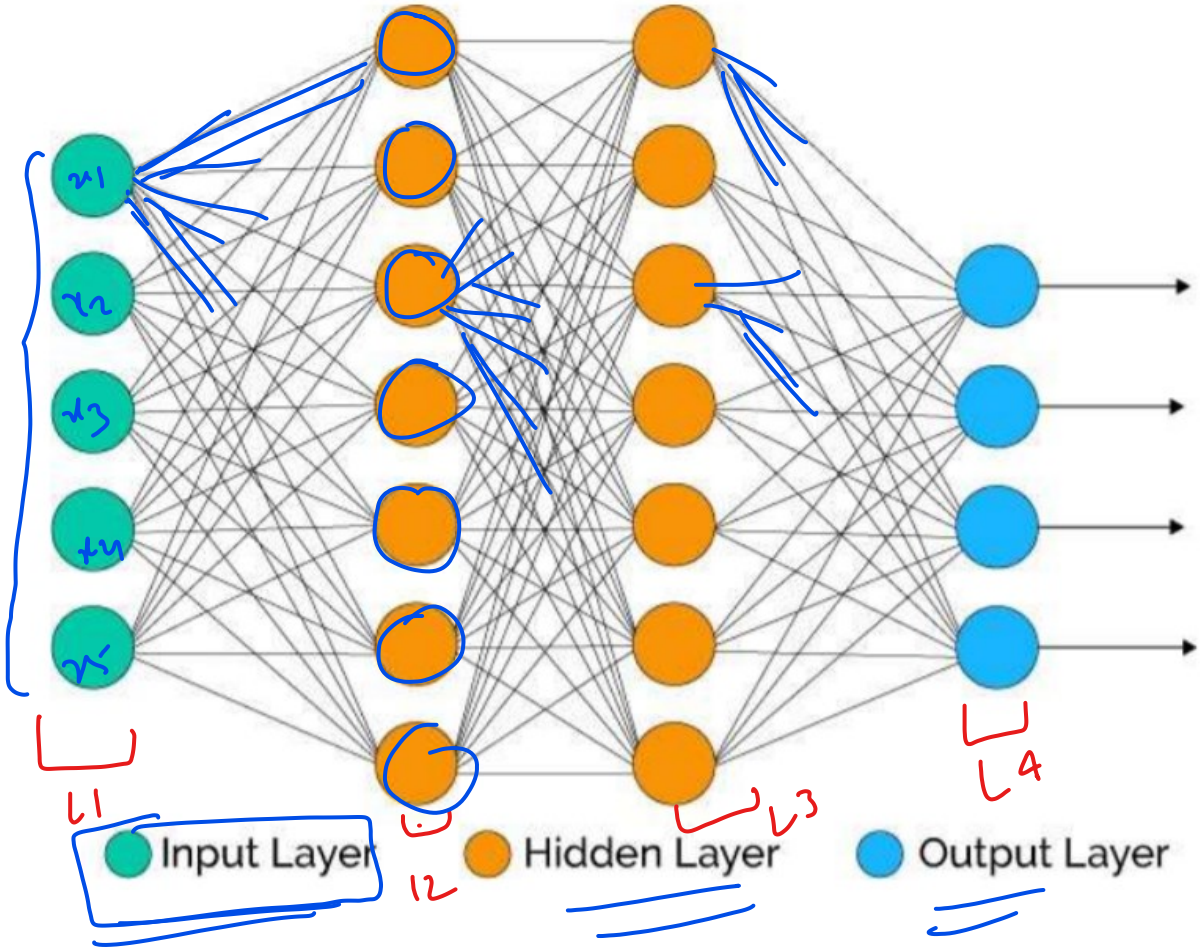
Fully connected networks

A typical neural network contains a large number of artificial neurons called units arranged in a series of layers.

① Neurons are arranged in layers

② num of neurons on i/p layer = num of features

③ num of neurons on o/p layer = num of classes.



$f(x) = x^2$

Two important rules in a neural network

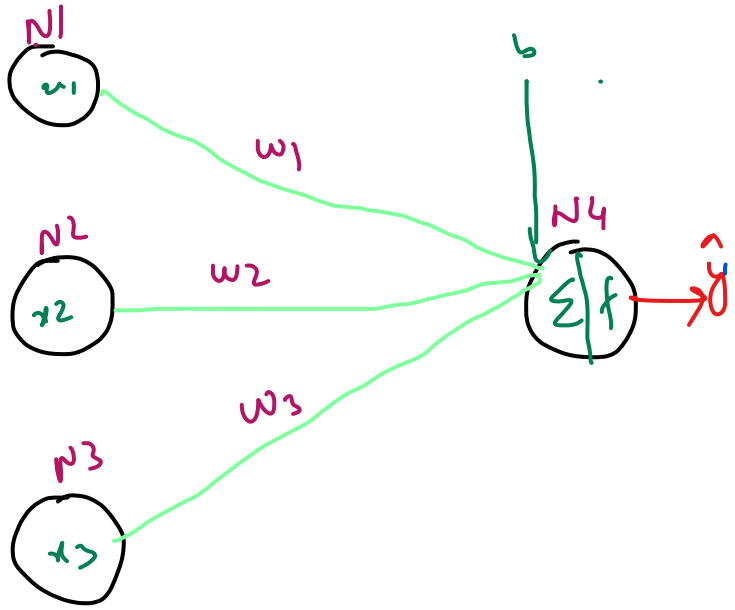
1. Input layer neurons do not perform any computations
2. Every computational neuron performs two steps:

   a. weighted sum with bias

   $$z = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

   b. Activation Function

   $$N_4 = f(z) = f(x_1 w_1 + x_2 w_2 + x_3 w_3 + b)$$



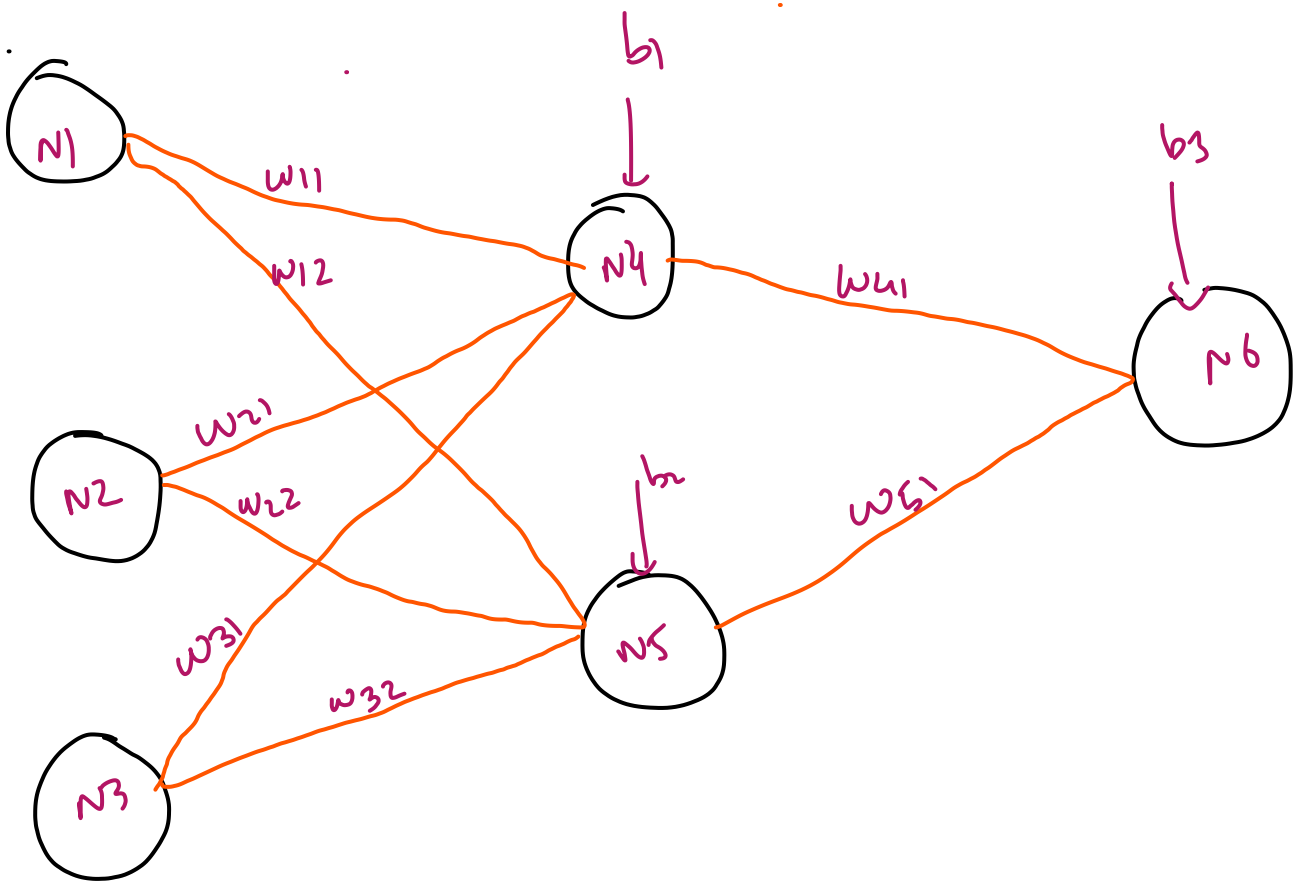
$$\hat{y} = N_4 = f(x_1 w_1 + x_2 w_2 + x_3 w_3 + b)$$

① AF = identity function $f(x) = x$ → Linear Regression - Regression

② AF = Sigmoid function $f(x) = \frac{1}{1+e^{-x}}$ → Logistic Reg. - Classification

L1 I/P

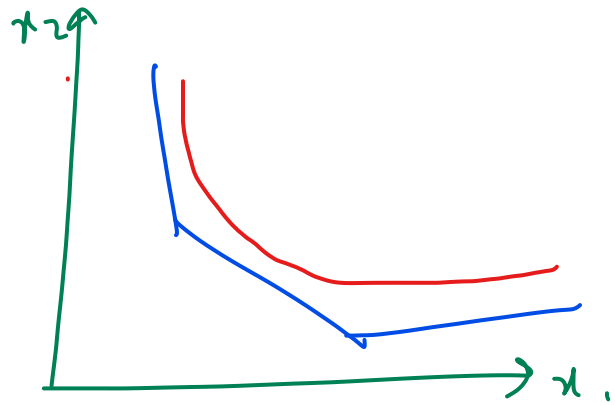
L2 O/P



$$N_4 = f(N_1 w_{11} + N_2 w_{21} + N_3 w_{31} + b_1)$$

$$N_5 = f(N_1 w_{12} + N_2 w_{22} + N_3 w_{32} + b_2)$$

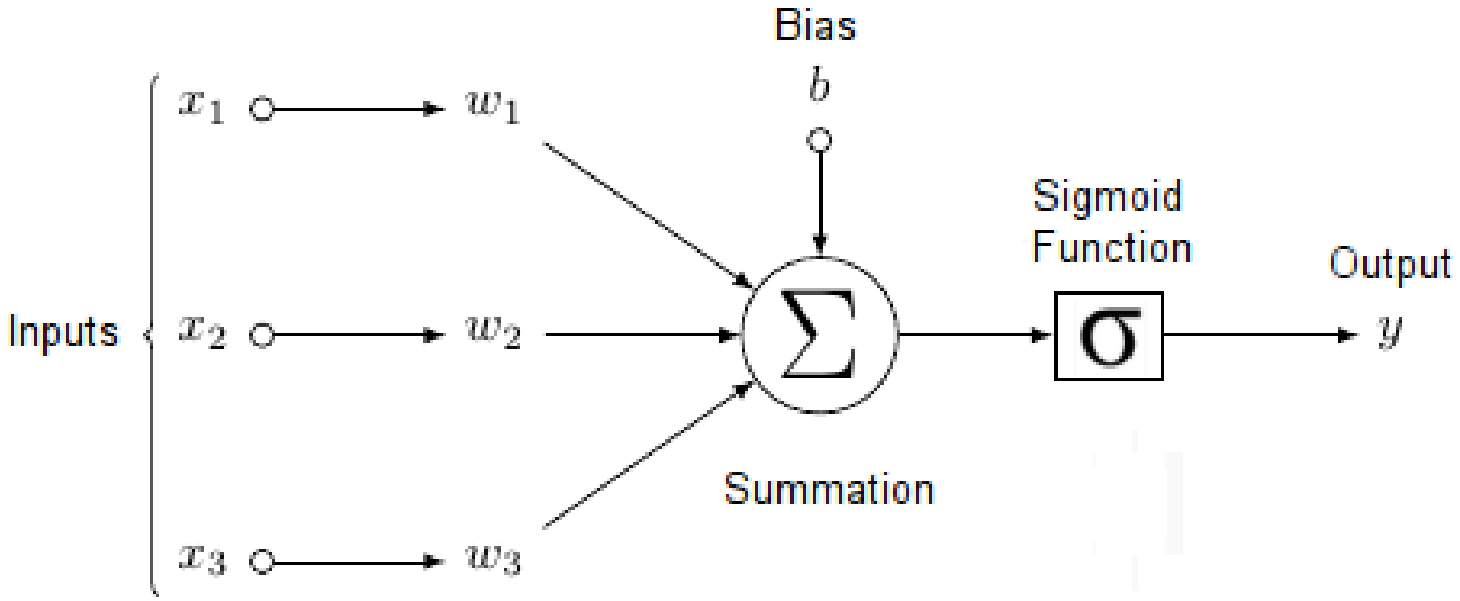
$$N_6 = f(N_4 w_{41} + N_5 w_{51} + b_3)$$



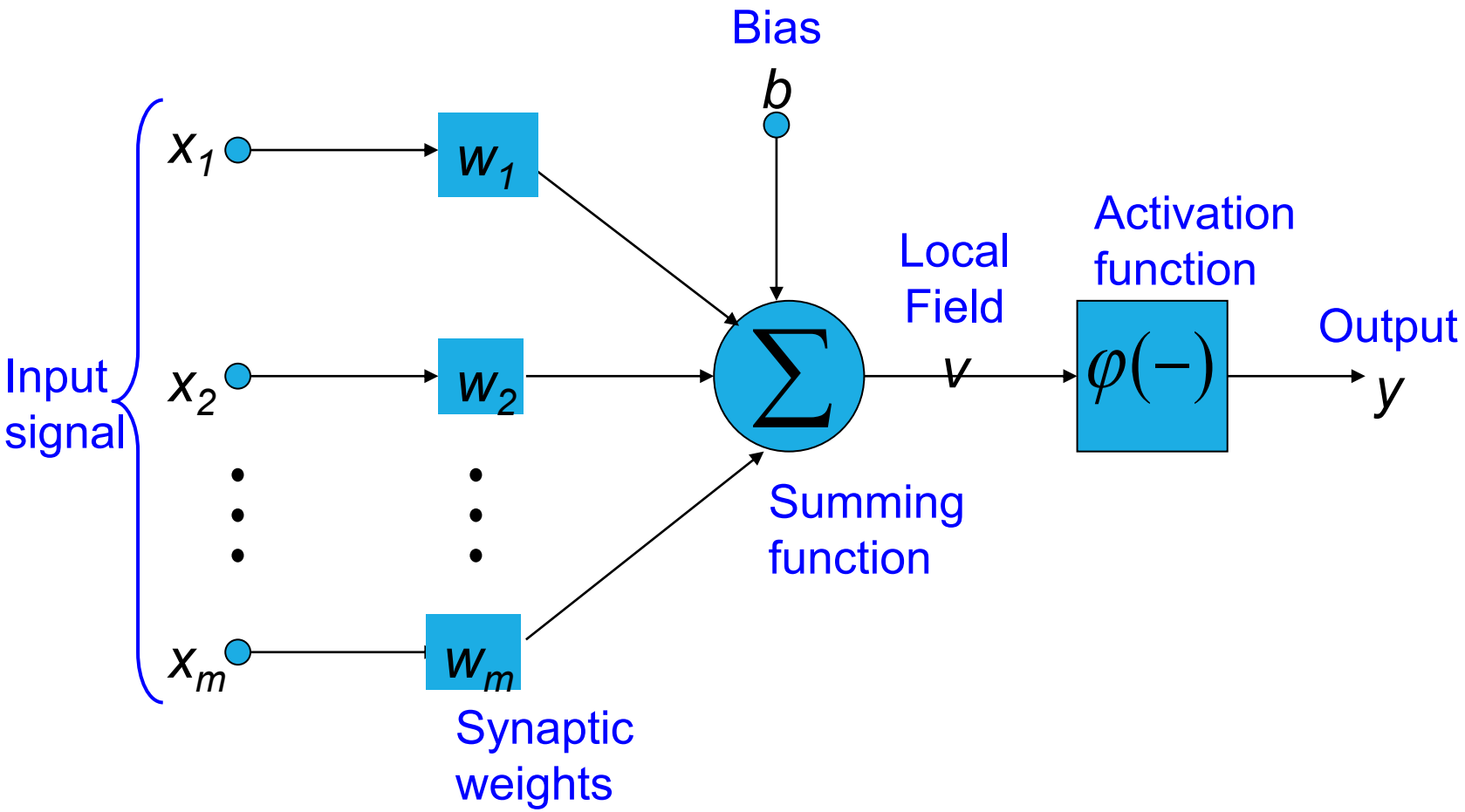
Elements of Neural Networks

1. No Computation on Input layer Neurons

2. Two process on every computational Neuron.
   - Weighted Sum
   - Activation Function



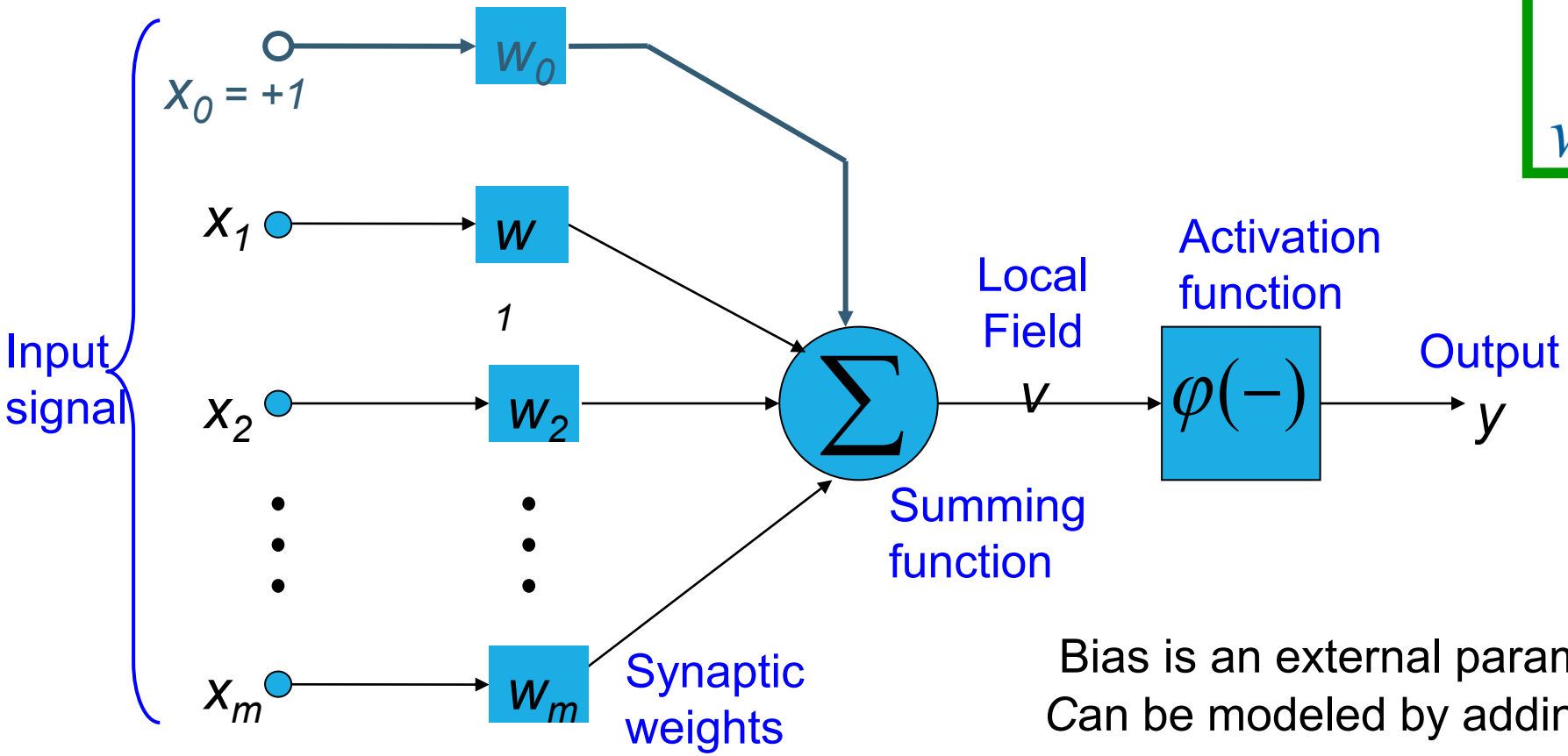
The Single Neuron Structure



Bias as extra input

$$v = \sum_{j=0}^{m} w_j x_j$$

$$w_0 = b$$



Bias is an external parameter of the neuron.
Can be modeled by adding an extra input.

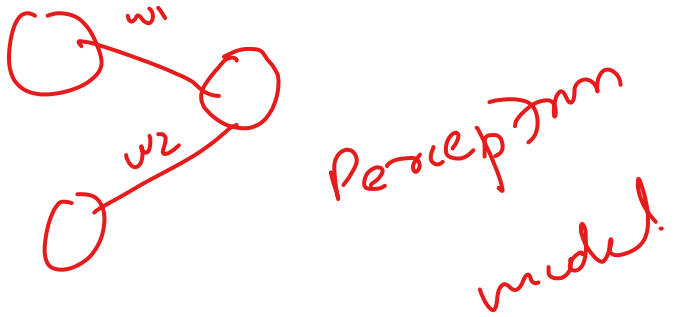
Network Architecture

Three basic different classes of network architectures

- Single-layer feed-forward Neural Networks
- Multi-layer feed-forward Neural Networks
- Recurrent Nerual Networks

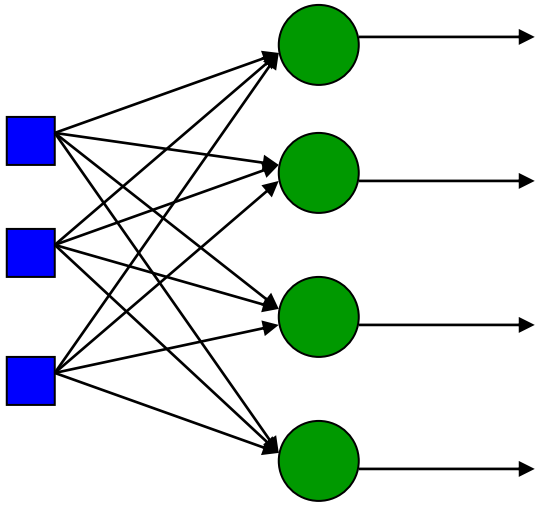
The architecture of a neural network is linked with the learning algorithm used to train

Single Layer Feed-forward



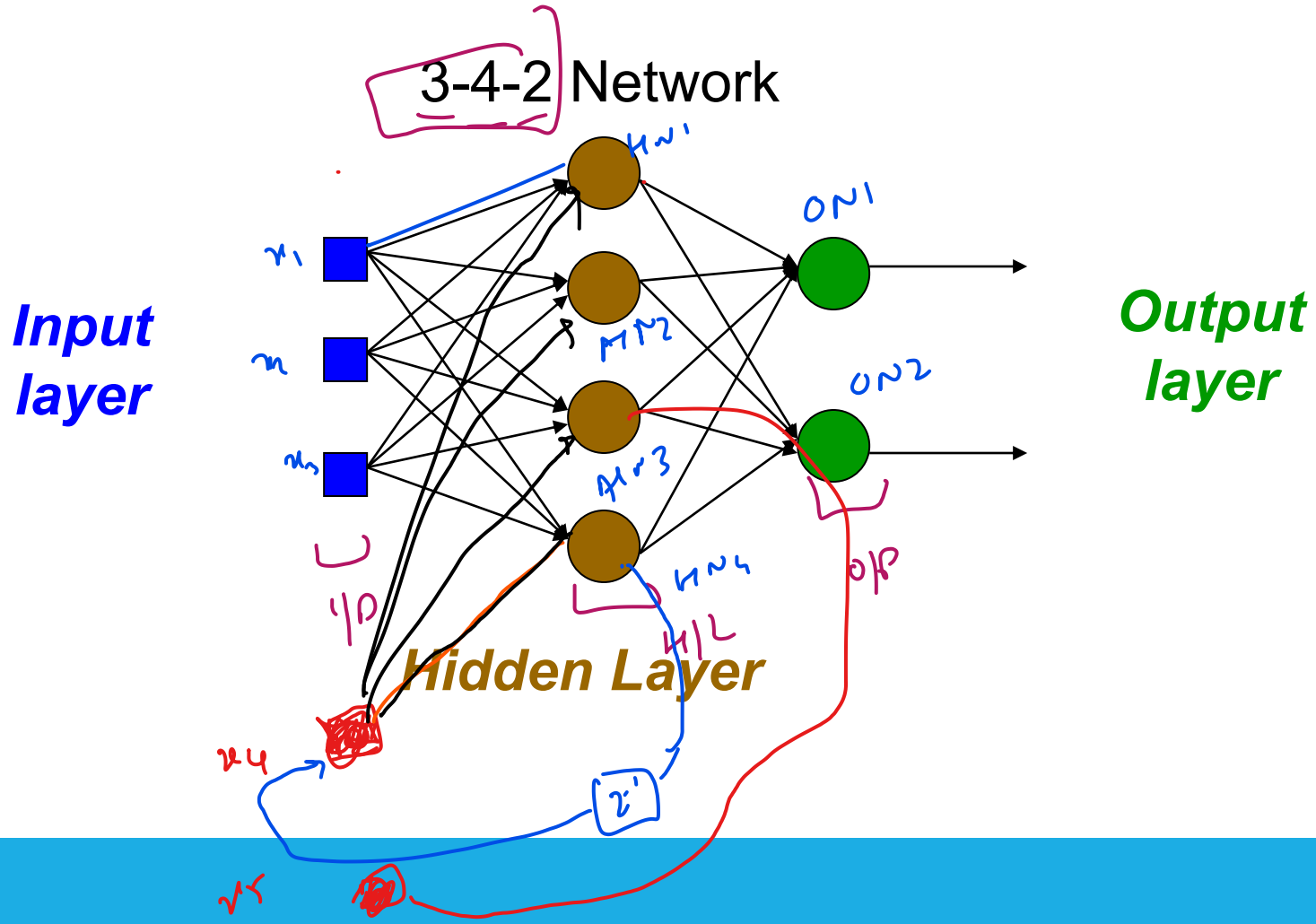
Neural Network having two input units and one output units with no hidden layers. These are also known as ‘single layer perceptrons.

Input layer of source nodes



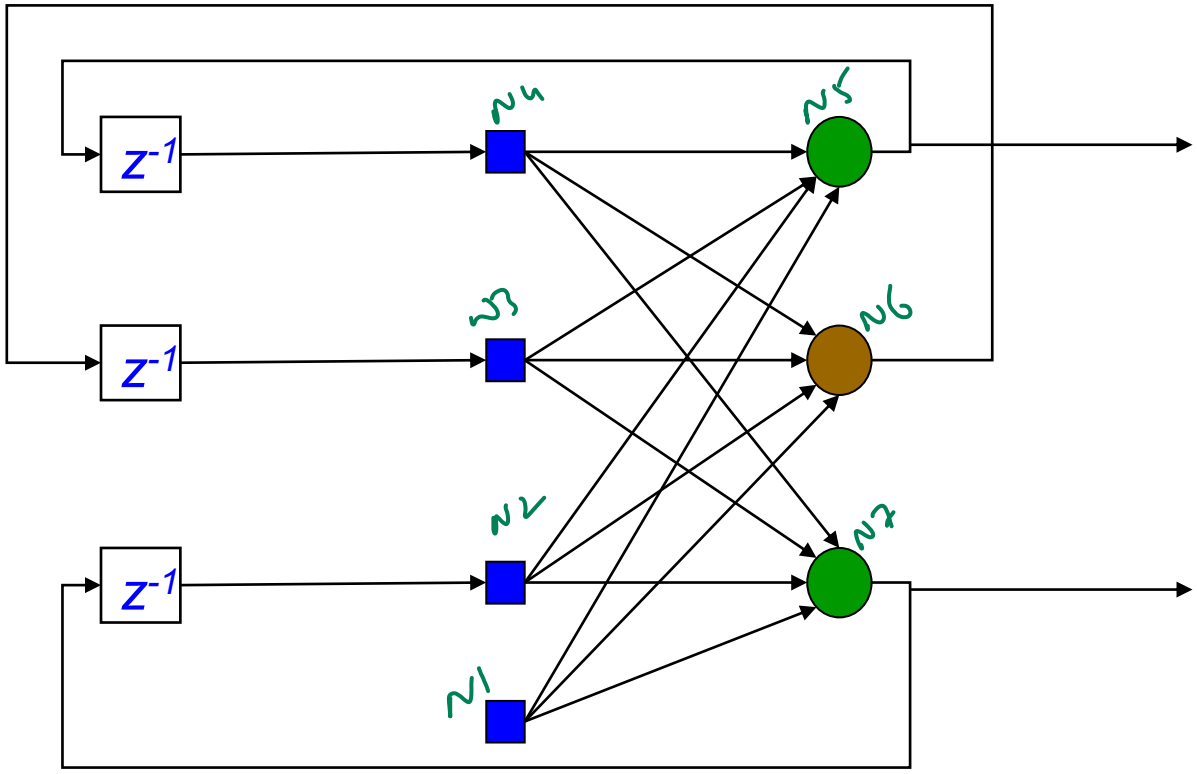
Output layer of neurons

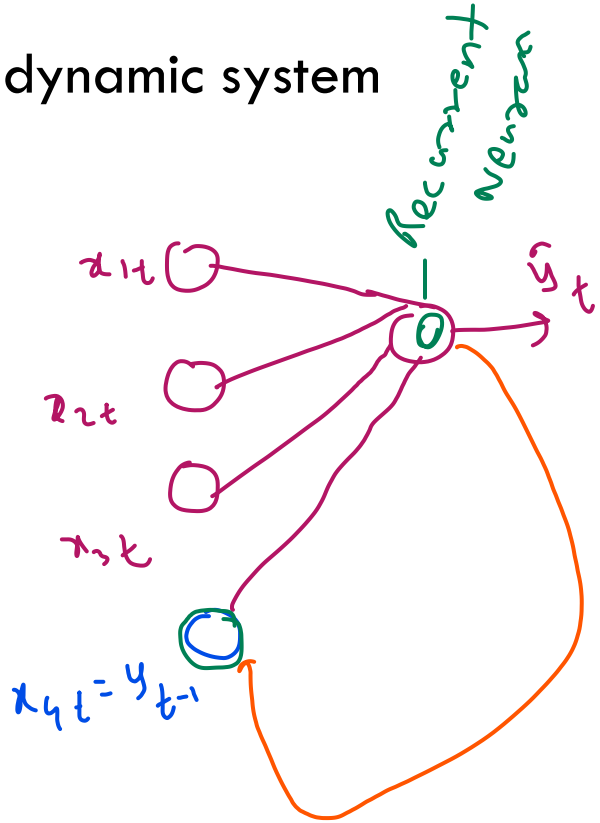
Multi Layer Feed-forward



Recurrent Network

Recurrent Network with ***hidden neuron(s)***: unit delay operator z^{-1} implies dynamic system





What are Activation Functions?

Activation Functions *introduce non-linear properties to Neural Network.*

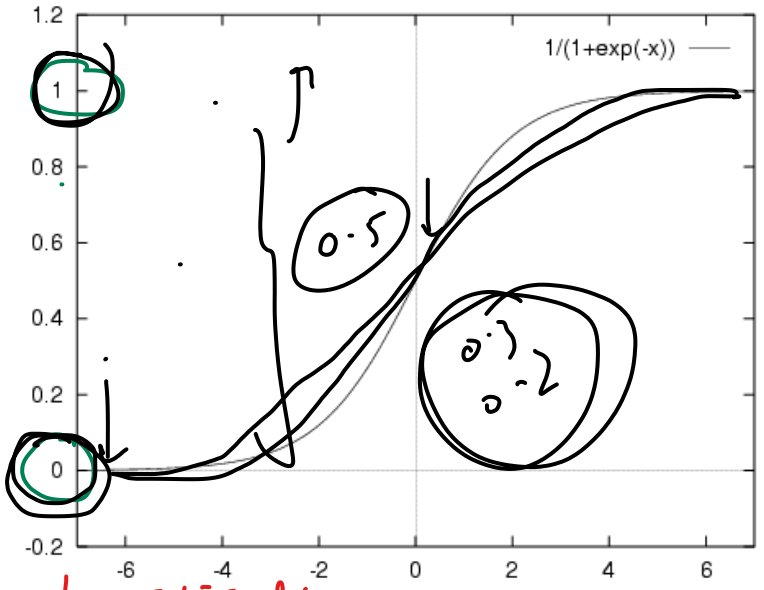
There are 2 types of activation functions –

1. Linear Activation Functions

2. Non Linear Activation Functions

Sigmoid Function

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

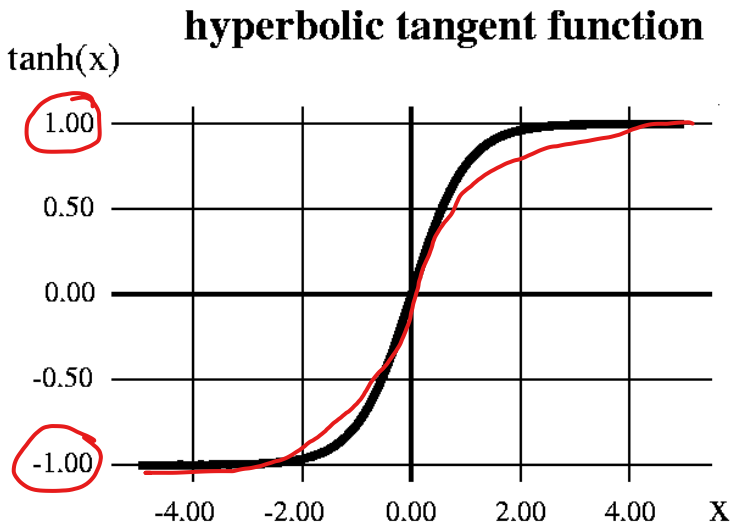


- The function is **differentiable**. That means, we can find the slope of the sigmoid curve at any two points.
- The function is **monotonic** but function's derivative is not.
- The logistic sigmoid function can cause a neural network to get stuck at the training time.
- The **softmax function** is a more generalized logistic activation function which is used for multiclass classification.

Hyperbolic Tangent function

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

- The function is **differentiable.**
- The function is **monotonic** while its **derivative is not.**
- The tanh function is mainly used classification between two classes.
- output is zero centered because its range in between -1 to 1 i.e -1 < output < 1 . Hence optimization is *easier* in this method hence in practice it is always preferred over Sigmoid function .



ReLu- Rectified Linear units

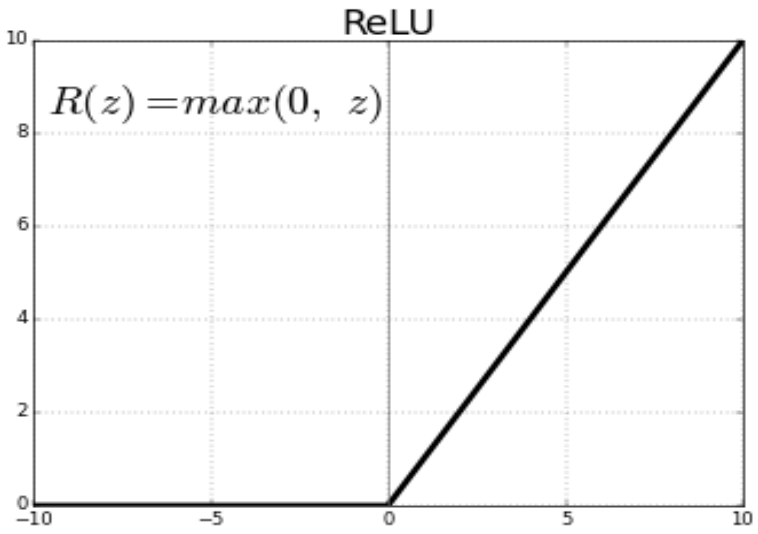
$$f(x) = \max(x, 0)$$

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

The function and its derivative **both are monotonic.**

It should only be used within Hidden layers of a Neural Network Model.

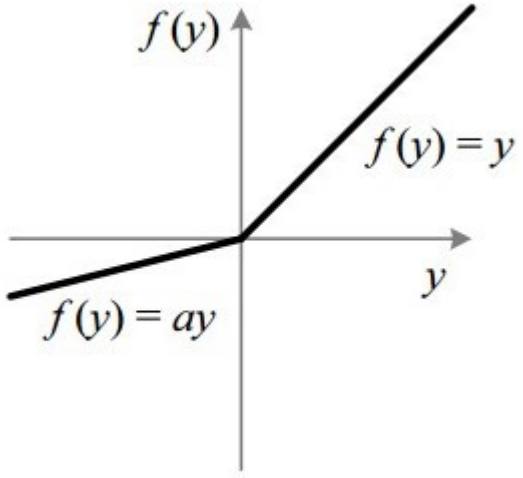
Hence for output layers we should use a **Softmax** function for a Classification problem to compute the probabilites for the classes , and for a regression problem it should simply use a **linear** function.

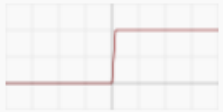
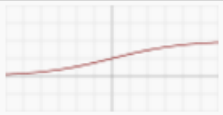
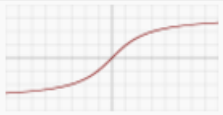


Leaky ReLu

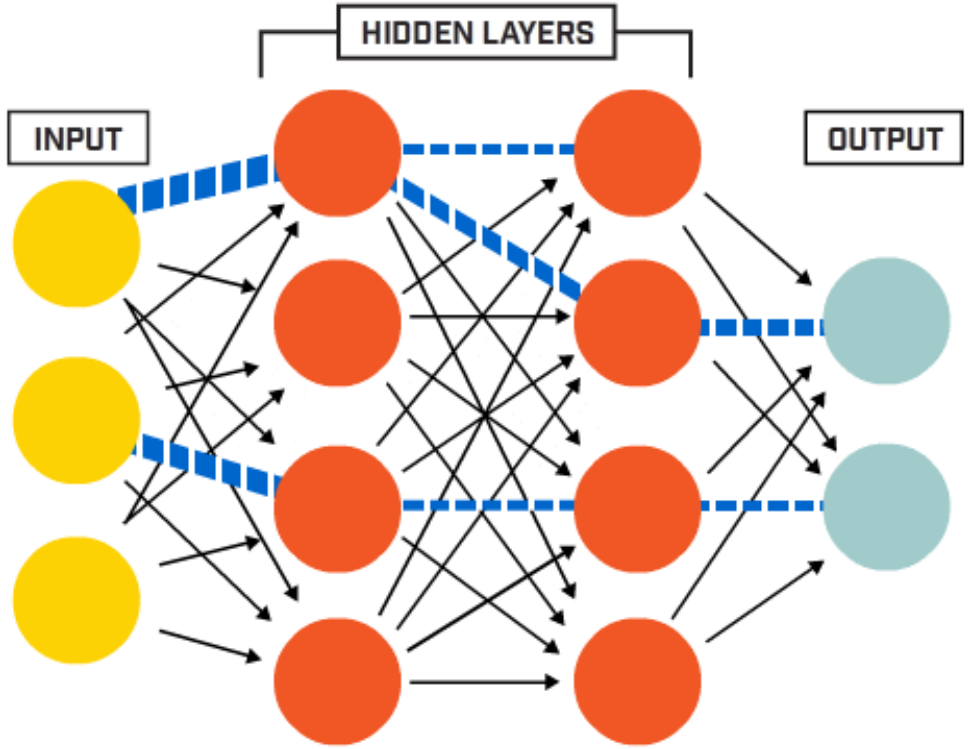
$$f(x) = \begin{cases} x & if\ x > 0 \\ 0.01x & otherwise \end{cases}$$

- The problem with ReLu is that some gradients can be fragile during training and can die. It can cause a weight update which will makes it never activate on any data point again. Simply saying that ReLu could result in Dead Neurons.
- To fix the problem of dying neurons another modification was introduced called Leaky ReLu. It introduces a small slope to keep the updates alive.

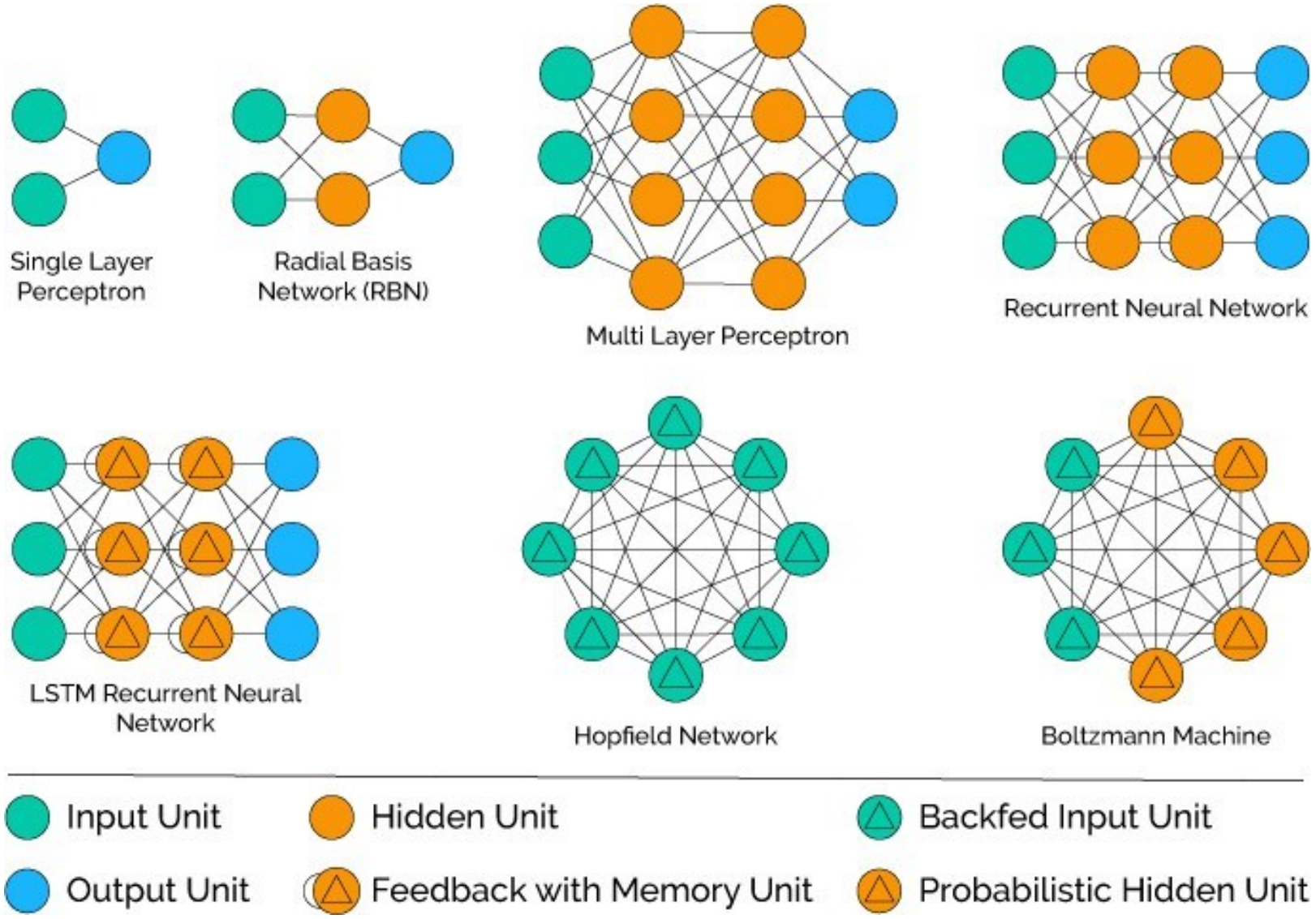


Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
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		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Network Architecture



Neural Network Types

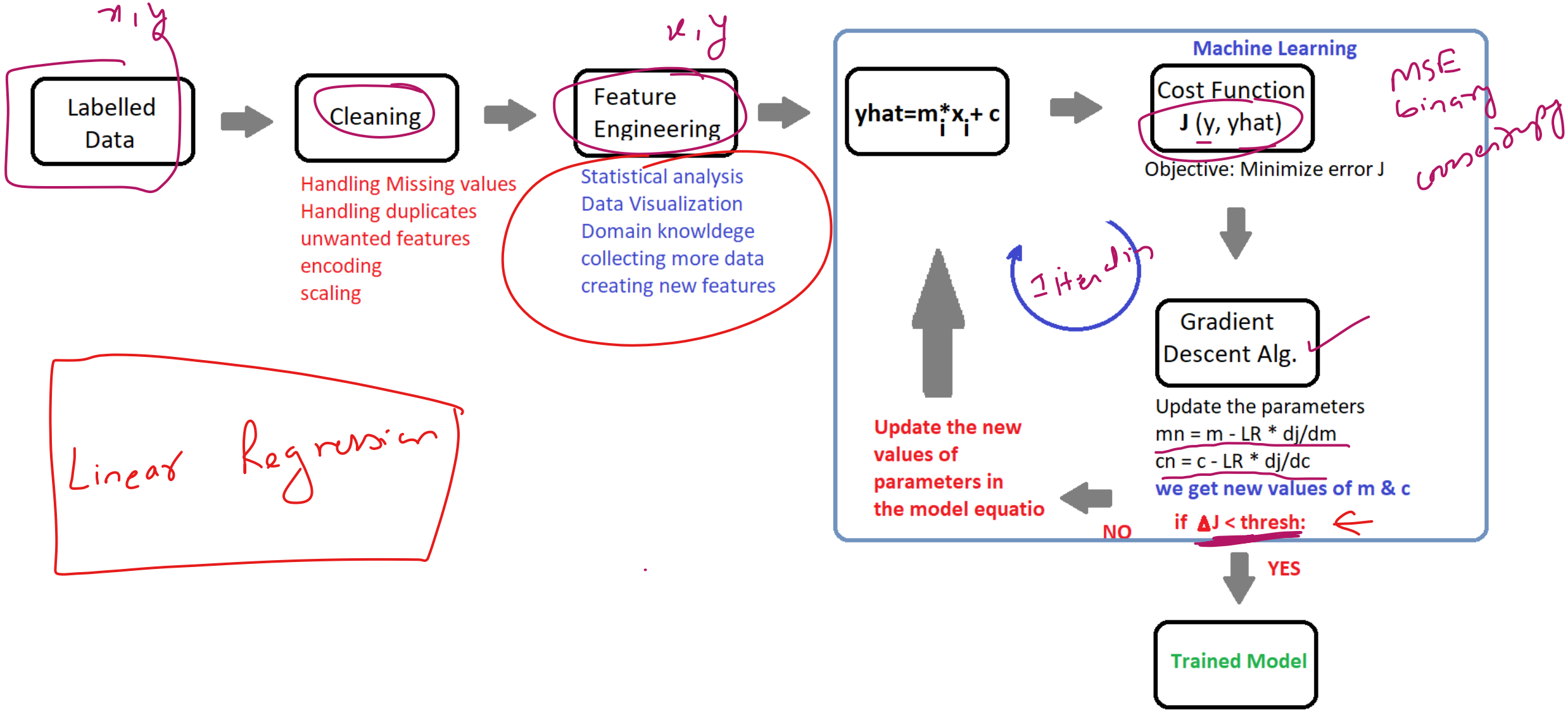


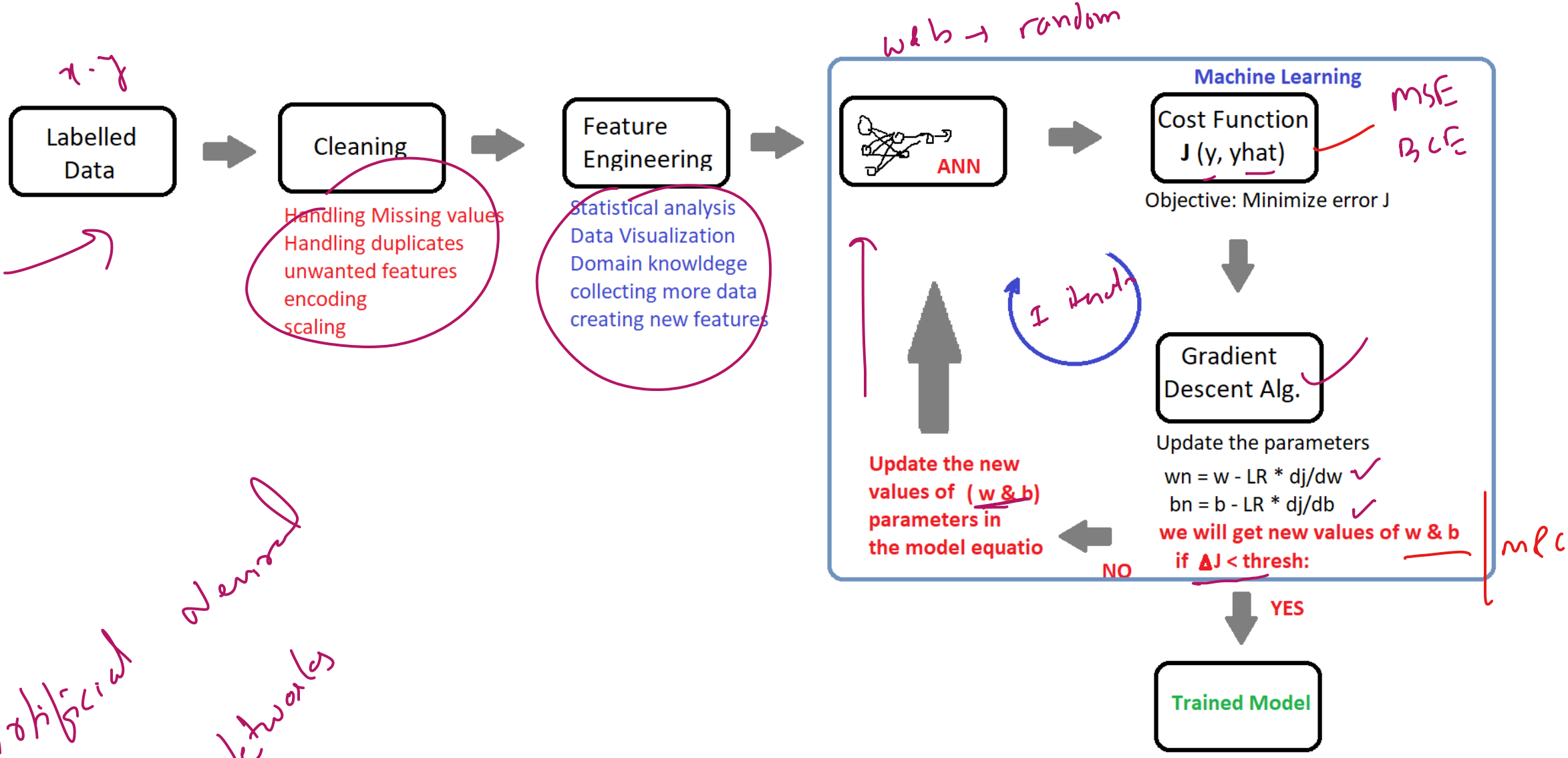
Neural Network for Machine Learning

- Multilayer Perceptron (supervised classification)
- Back Propagation Network (supervised classification)
- Hopfield Network (for pattern association)
- Deep Neural Networks (unsupervised clustering)

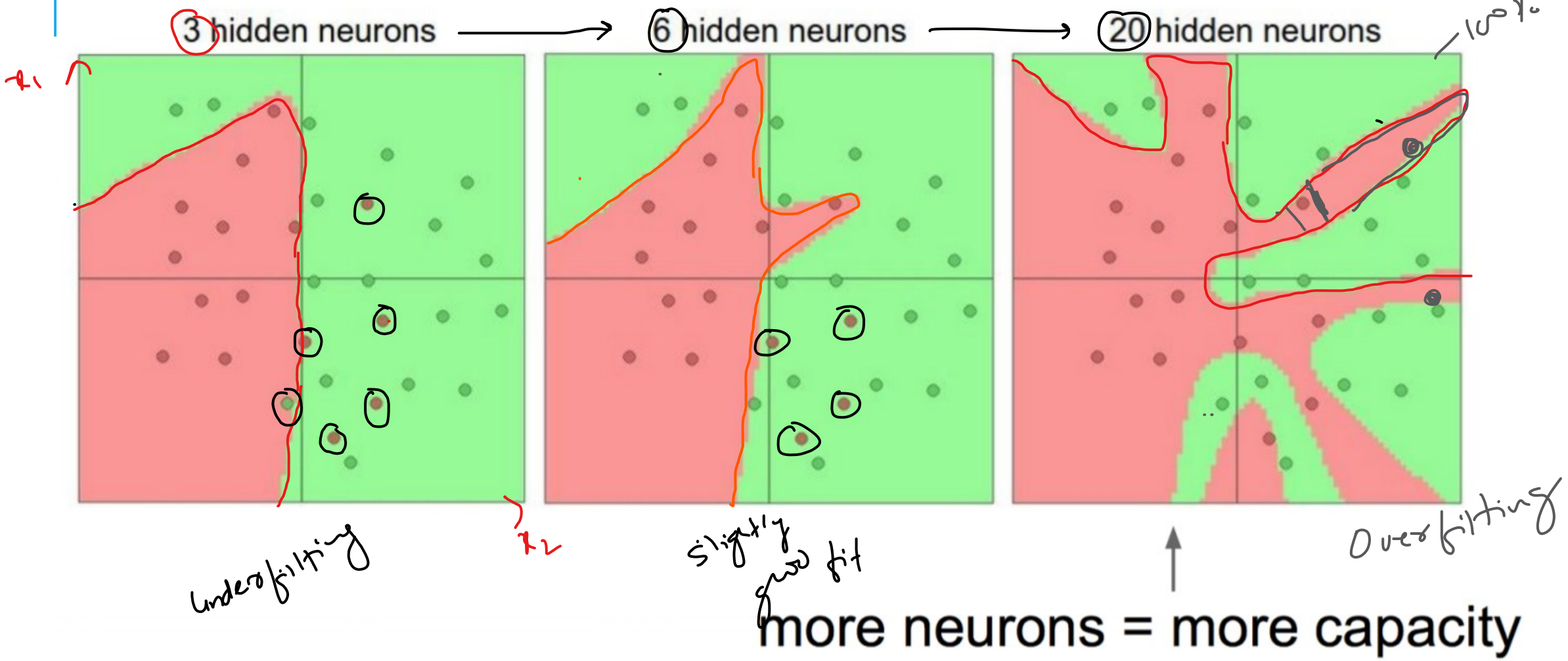
Neural Network for Deep Learning

- Recurrent neural network
- Multi-layer perceptrons (MLP)
- Convolutional neural networks
- Recursive neural networks
- Deep belief networks
- Convolutional deep belief networks
- Self-Organizing Maps
- Deep Boltzmann machines
- Stacked de-noising auto-encoders


Labelled Data
Cleaning
Handling Missing values
Handling duplicates
unwanted features
encoding
scaling
Feature Engineering
Statistical analysis
Data Visualization
Domain knowldege
collecting more data
creating new features
Machine Learning
yhat=m*x+ c
Cost Function
J (y, yhat)
Objective: Minimize error J
Gradient Descent Alg.
Update the parameters
mn = m - LR * dj/dm
cn = c - LR * dj/dc
we get new values of m & c
if ΔJ < thresh:
NO
YES
Update the new values of parameters in the model equatio
Trained Model
Linear Regression


Labelled Data
Cleaning
Handling Missing values
Handling duplicates
unwanted features
encoding
scaling
Feature Engineering
Statistical analysis
Data Visualization
Domain knowldege
collecting more data
creating new features
Machine Learning
ANN
Cost Function
J (y, yhat)
Objective: Minimize error J
Gradient Descent Alg.
Update the parameters
wn = w - LR * dj/dw
bn = b - LR * dj/db
we will get new values of w & b
if ΔJ < thresh:
NO
Update the new values of (w & b) parameters in the model equatio
YES
Trained Model

How many hidden Neurons?



Advantages of Neural Networks

- A neural network can perform tasks that a linear program can not.
- When an element of the neural network fails, it can continue without any problem by their parallel nature.
- A neural network learns and does not need to be reprogrammed.
- It can be implemented in any application.
- It can be performed without any problem.

Limitations of Neural Networks

- The neural network needs the training to operate.
- The architecture of a neural network is different from the architecture of microprocessors, therefore, needs to be emulated.
- Requires high processing time for large neural networks.

Important terminologies

- **Feature:** The input(s) to our model
- **Examples:** An input/output pair used for training
- **Labels:** The output of the model
- **Layer:** A collection of nodes connected together within a neural network.
- **Model:** The representation of your neural network
- **Dense and Fully Connected (FC):** Each node in one layer is connected to each node in the previous layer.
- **Weights and biases:** The internal variables of model
- **Loss:** The discrepancy between the desired output and the actual output
- **MSE:** Mean squared error, a type of loss function that counts a small number of large discrepancies as worse than a large number of small ones.

Important terminologies

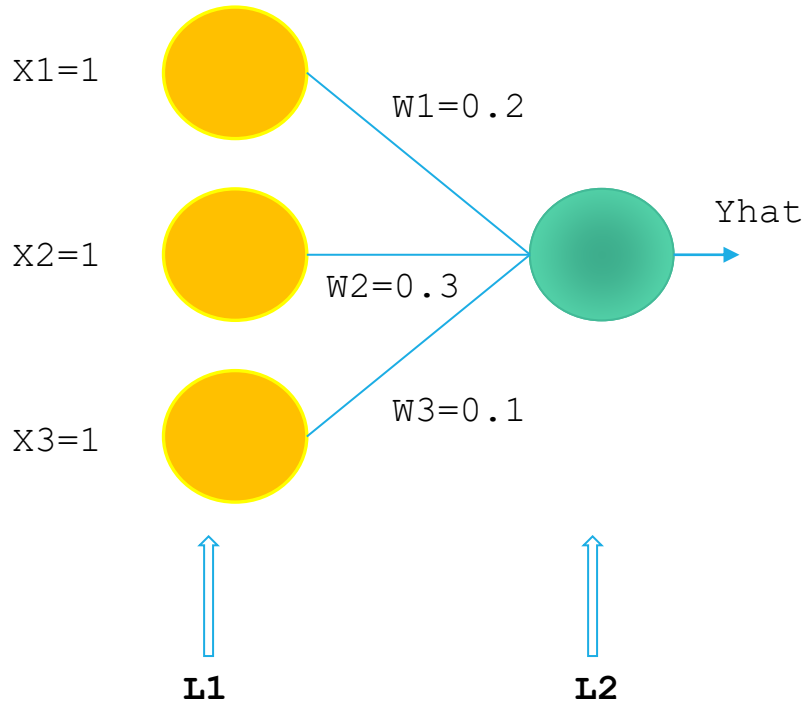
- **Gradient Descent:** An algorithm that changes the internal variables a bit at a time to gradually reduce the loss function.
- **Optimizer:** A specific implementation of the gradient descent algorithm. (There are many algorithms for this. In this course we will only use the "Adam" Optimizer, which stands for *ADAptive with Momentum*. It is considered the best-practice optimizer.)
- **Learning rate:** The "step size" for loss improvement during gradient descent.
- **Batch:** The set of examples used during training of the neural network
- **Epoch:** A full pass over the entire training dataset
- **Forward pass:** The computation of output values from input
- **Backward pass (backpropagation):** The calculation of internal variable adjustments according to the optimizer algorithm, starting from the output layer and working back through each layer to the input.

Applications of Neural Networks

Application	Architecture / Algorithm	Activation Function
Process modeling and control	Radial Basis Network	Radial Basis
Machine Diagnostics	Multilayer Perceptron	Tan- Sigmoid Function
Portfolio Management	Classification Supervised Algorithm	Tan- Sigmoid Function
Target Recognition	Modular Neural Network	Tan- Sigmoid Function
Medical Diagnosis	Multilayer Perceptron	Tan- Sigmoid Function
Credit Rating	Logistic Discriminant Analysis with ANN, Support Vector Machine	Logistic function

Single Layer Neural Network

```
# Single layer Neural Network
# x1,x2=,x3=1,1,1
# w1,w2,w3=0.2,0.3,0.1
# Activation function --> fx=2*x
```

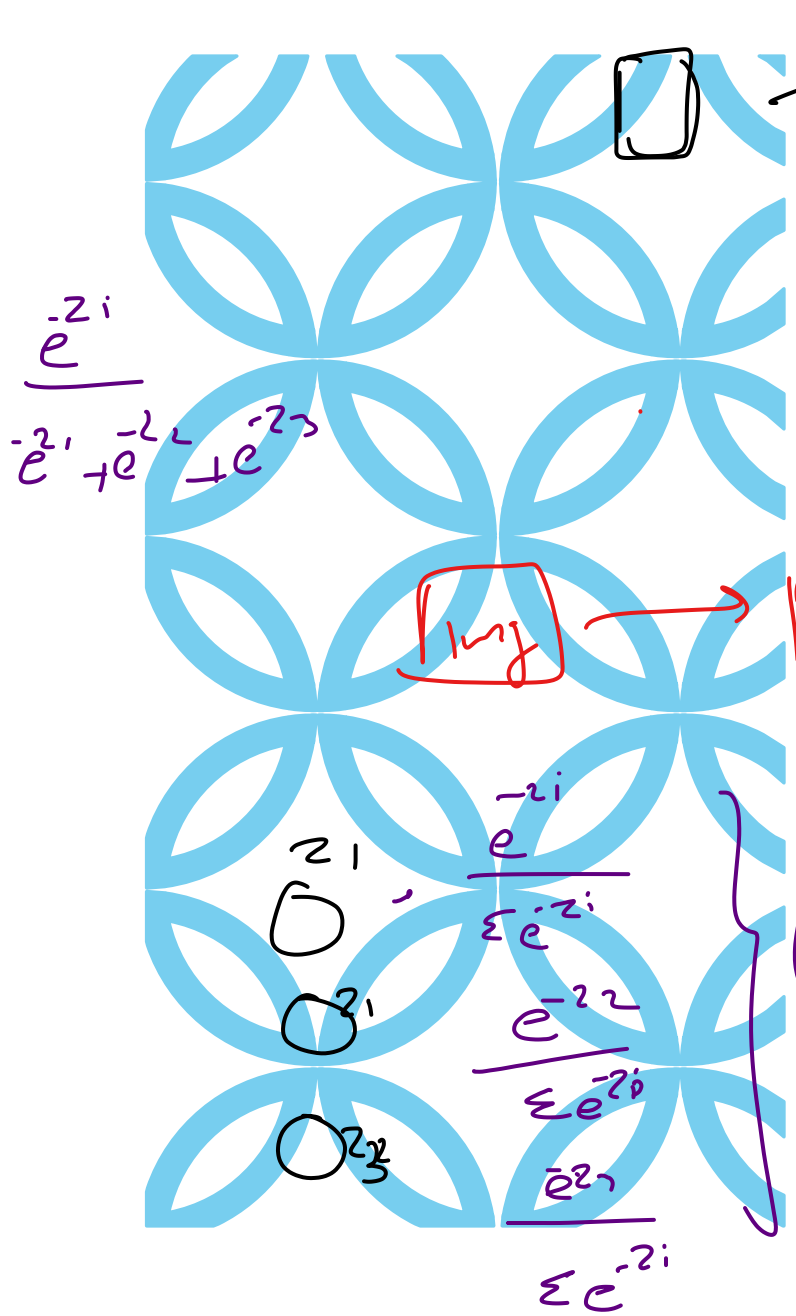


Single Layer Neural Network

```
import numpy
x=numpy.array([1.0,1.0,1.0]).reshape(1,-1)
t=numpy.array([30.0])
w=numpy.array([0.2,0.3,0.1]).reshape(3,1)
```

Single Layer Neural Network

```
for i in range(30):
    l1=x #loading the data into the network - input layer
    #weighted sum of inputs from l1 layer & weights
    z=numpy.dot(l1,w)
    l2=2*z #activation function
    #GDA - wn=w-LR*dj/dw
    djdw=(l2-t)*2*l1
    w=w-0.01*djdw.T
    print(l2)
print(w)
```



Stay tuned for Model Representation and Backpropagation Algorithm

THANK YOU