Public Notice regarding Use of Images

This document contains images obtained by routine Google Images searches. Some of these images may perhaps be under copyright. They are included here for educational and noncommercial purposes and are considered to be covered by the doctrine of Fair Use. In any event they are easily available from Google Images.

It's not feasible to give full scholarly credit to the creators of these images. We hope they can be satisfied with the positive role they are playing in the educational process.

AI and Deep Learning Skilling and Research

Prerequisites

- Interest in Deep Learning and AI
- One Programming Language
- Commitment to Persevere

Motivation for the

It is redefining the way computing is being used to solve societal problems

Some of the existing jobs are going to vanish or change and most attractive jobs are in deep learning

It is one of fastest growing technology being adapted across disciplines

Most of the startups now have an element of AI to build new applications

Objectives and Outcomes

Knowledge of deep learning concepts

To apply appropriate tuning to improve the accuracy of network

To enable you to develop sample projects

Course Coverage

- Foundational Concepts and Machine Learning Basics
- Deep Learning Fundamentals, Models and Intuition
- Convolutional Neural Networks
- Recurrent Neural Networks

References and Resources

Online Courses of Machine Learning, Deep Learning and Data Science on edX, Udacity, Coursera

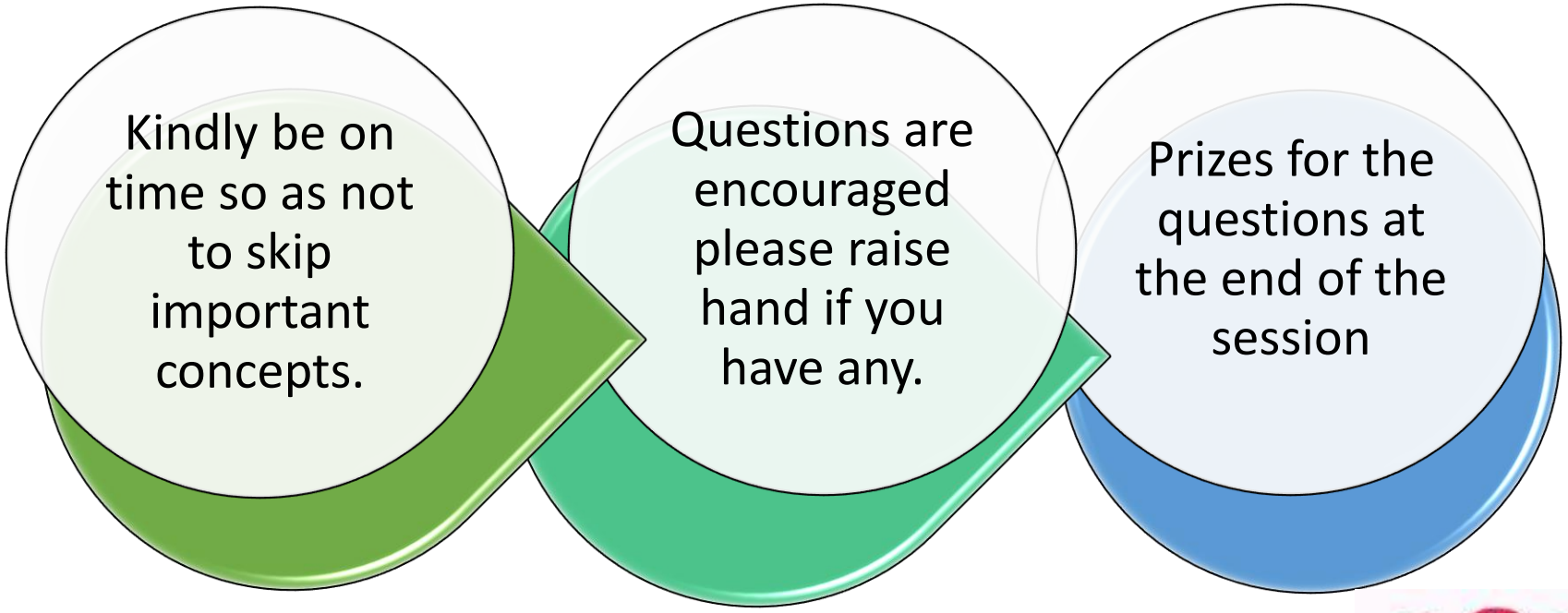
Series of Courses by Andrew Ng

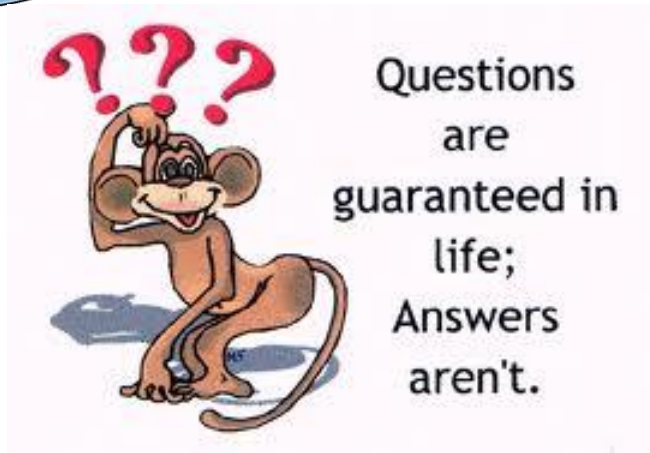
GitHub Code and sample projects provided by many programmers

Blogs and Articles on the latest issues and trends in Deep Learning

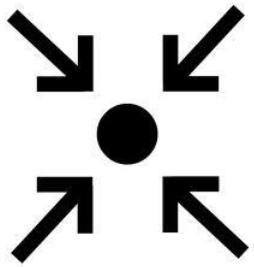
Resources page of leadingindia.ai

Administration





Session Coverage



Leadingindia.ai

Applications of Deep Learning

Why it has become so popular now?

Foundational concepts

Python Preliminaries

Linear and Logistic Regression

NVIDIA DGX-1 V100 Super Computer 1st in India arrived at our lab in Nov 2017

AICTE Chairman Anil Sahasrabudhe Inaugurating leadingindia.ai website at Bennett University Campus

Niti Ayog Chief Amitabh Kant Inaugurating Amazon Intelligent Devices Lab at Bennett University

Making Deep Learning and AI skills mainstream in India to fulfill trilateral needs of entrepreneurship, Industry academia partnership and application-inspired Engineering Research

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

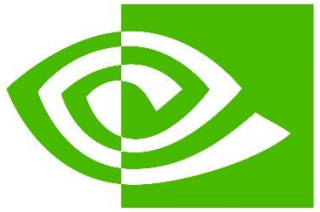
Brunel University London

UCL

Newton-Bhabha Fund

ROYAL ACADEMY OF ENGINEERING

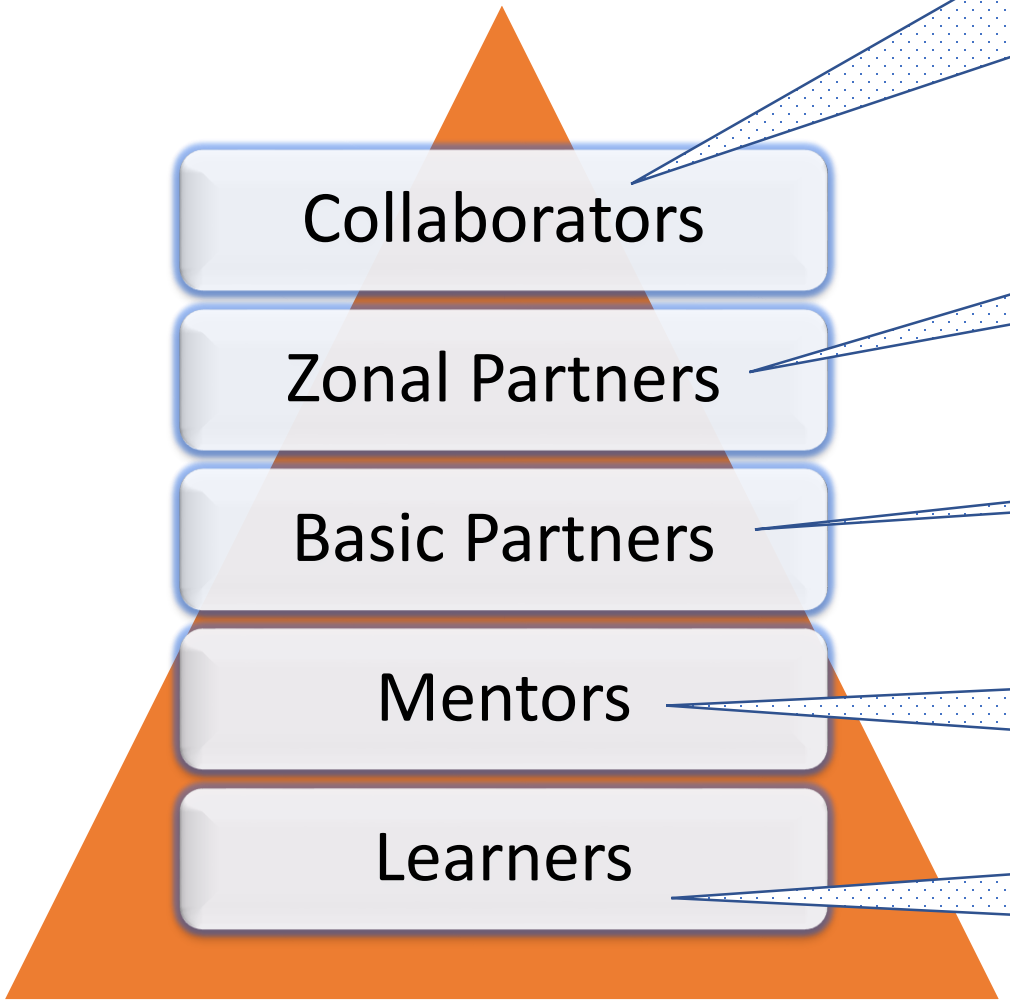
IEEE computer society



NVIDIA

powered by aws educate

Level of Associates



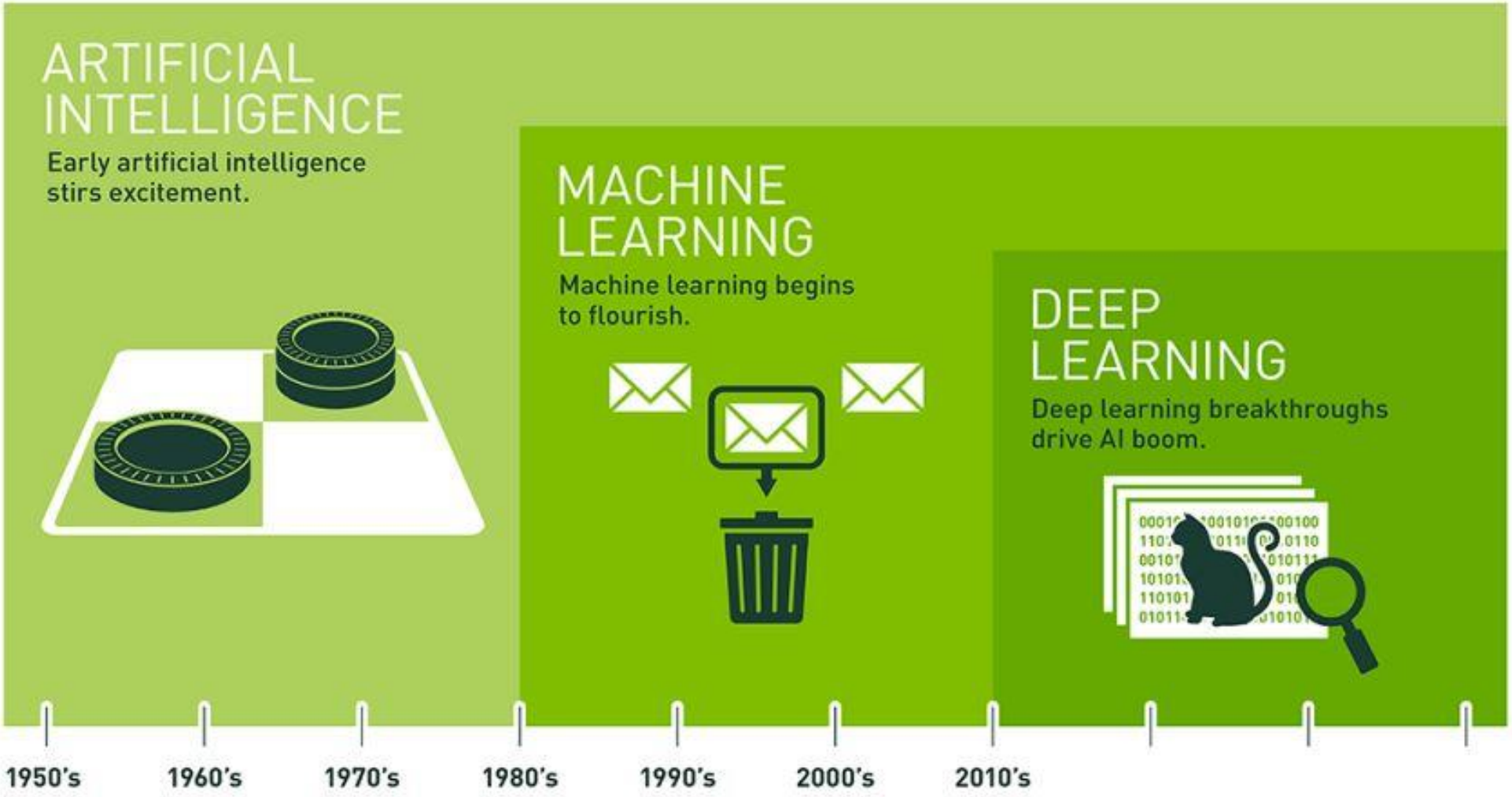
Those who want to be seen as leaders and resource institutions for Deep Learning and AI Research

Those who want to flourish with quality research groups along with Skilling in Deep learning and AI

To create a pair of instructors to teach AI and Deep Learning in the institutions

Experts with experience to mentor research groups on AI

Individuals who want to learn and build their career in Deep Learning



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Deep Learning/AI APPLICATIONS

Few Popular Applications: Precision Agriculture, Learner Profiling, Video Captioning, Exploring Patterns from Satellite images, Image detection in Healthcare, Identifying specific markers in Genomes, Creating Art and Music, Recommendations, behavior prediction,

Three main areas where Deep learning is being prominently applied

Detection

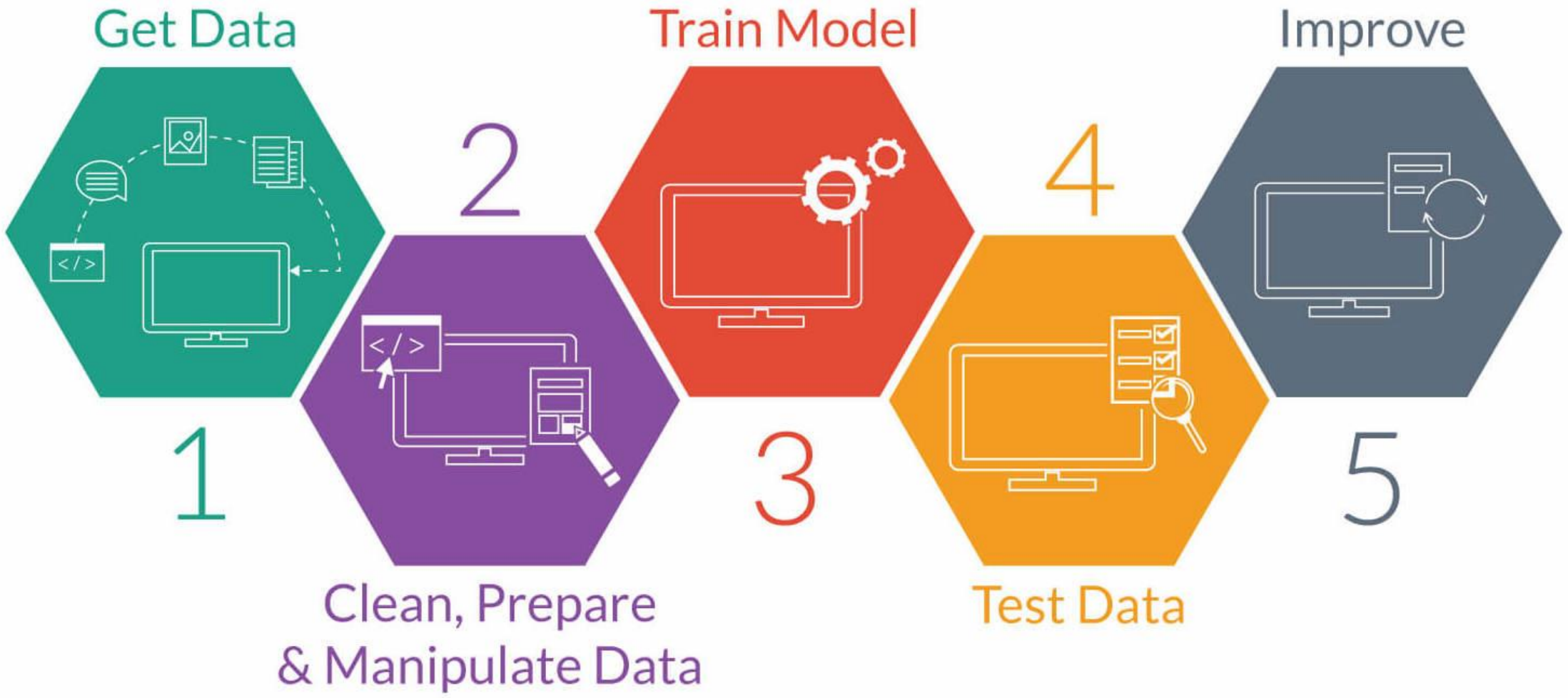
- Text & Speech
- Image interpretation
- Human behavior & identity
- Abuse & Fraud

Prediction

- Recommendations
- Individual behavior & condition
- Collective behavior

Generation

- Visual art
- Music
- Text
- Design


Get Data
Train Model
Improve
2
4
1
3
5
Clean, Prepare
& Manipulate Data
Test Data

1. Problems where a) There is no deterministic algorithm (not even of evil complexity) e.g. Recognizing a 3D object from a given scene, Handwriting recognition, Speech recognition

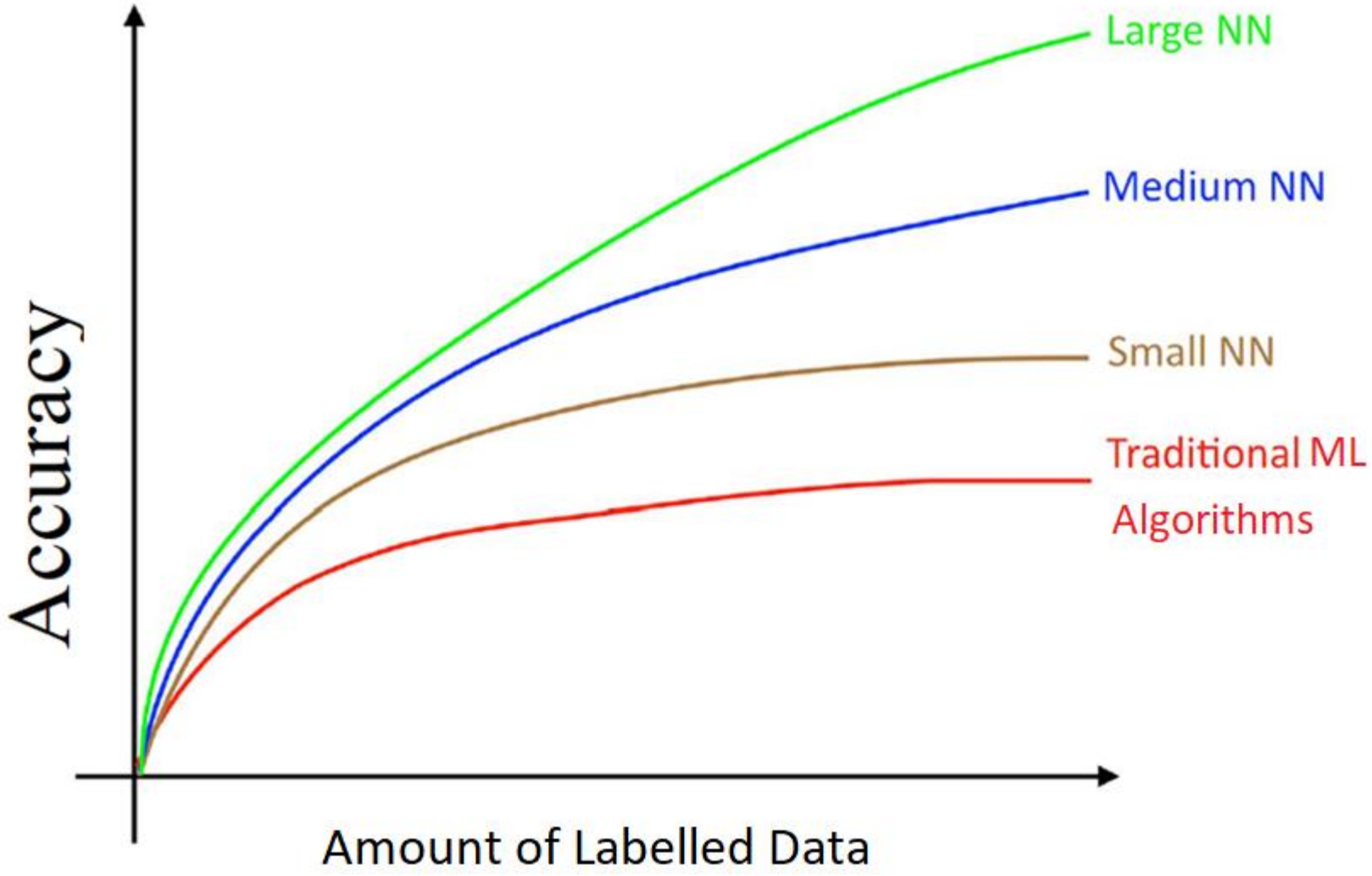
2. Problems which don't have a fix solution and goal posts keep changing. System adapts and learns from experience e.g. SPAM emails, Financial fraud, IT Security Framework

3. Where Solutions are Individual specific or time dependent. e.g. recommendations and targeted advertisements

4. For prediction based on past and existing patterns (not defined or defined by huge number of weak rules) e.g. prediction of share prices etc.

For What kind of Applications we use Machine/Deep Learning

Why it has taken off now



Availability of Data has increased due to explosion in Smart Mobiles and devices

More Computing Power is available due to coming of NVIDIA GPUs

Release/development of new algorithms, APIs and Platforms for Deep Learning Applications

Essential Tools

- Git and Github
- Python
- Jupyter Notebooks
- Numpy for Matrix and Vector calculations
- Pandas for Data handling, curation and manipulation
- Matplotlib for plotting of graphs for different analysis on data
- So many other Python APIs to make your life easy to develop DL models

Reasons why developers love Python

It's simple

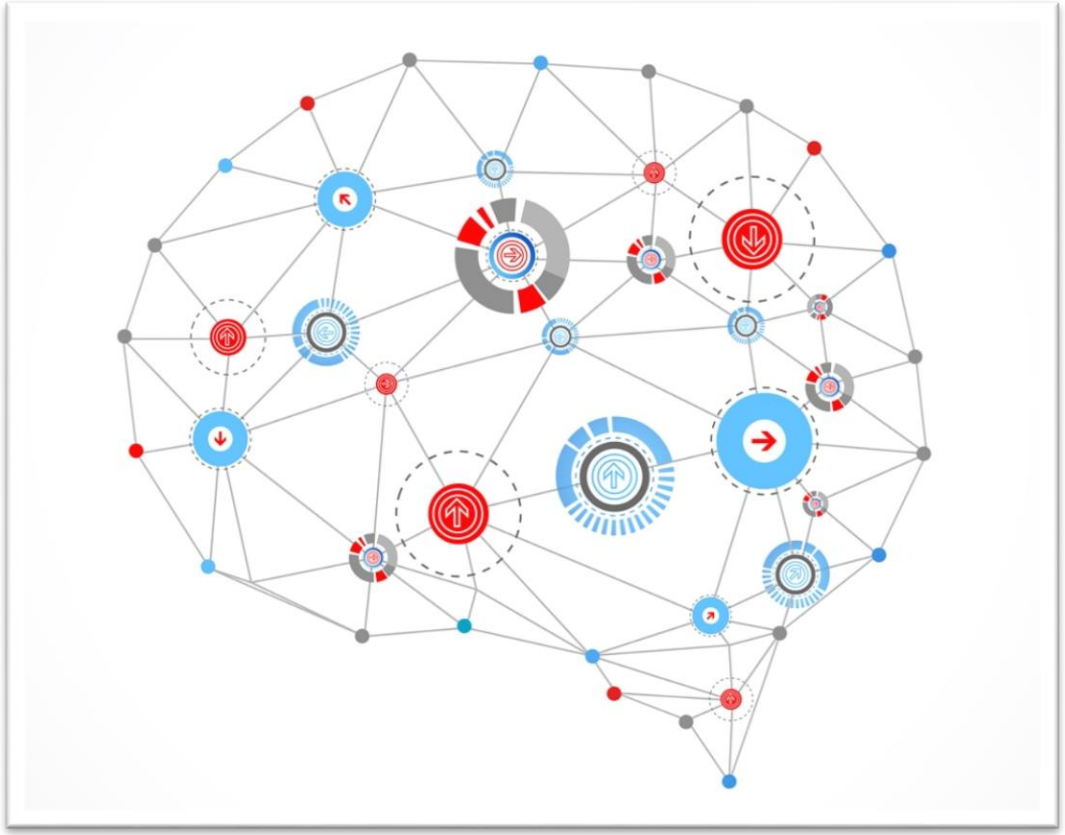
It's free

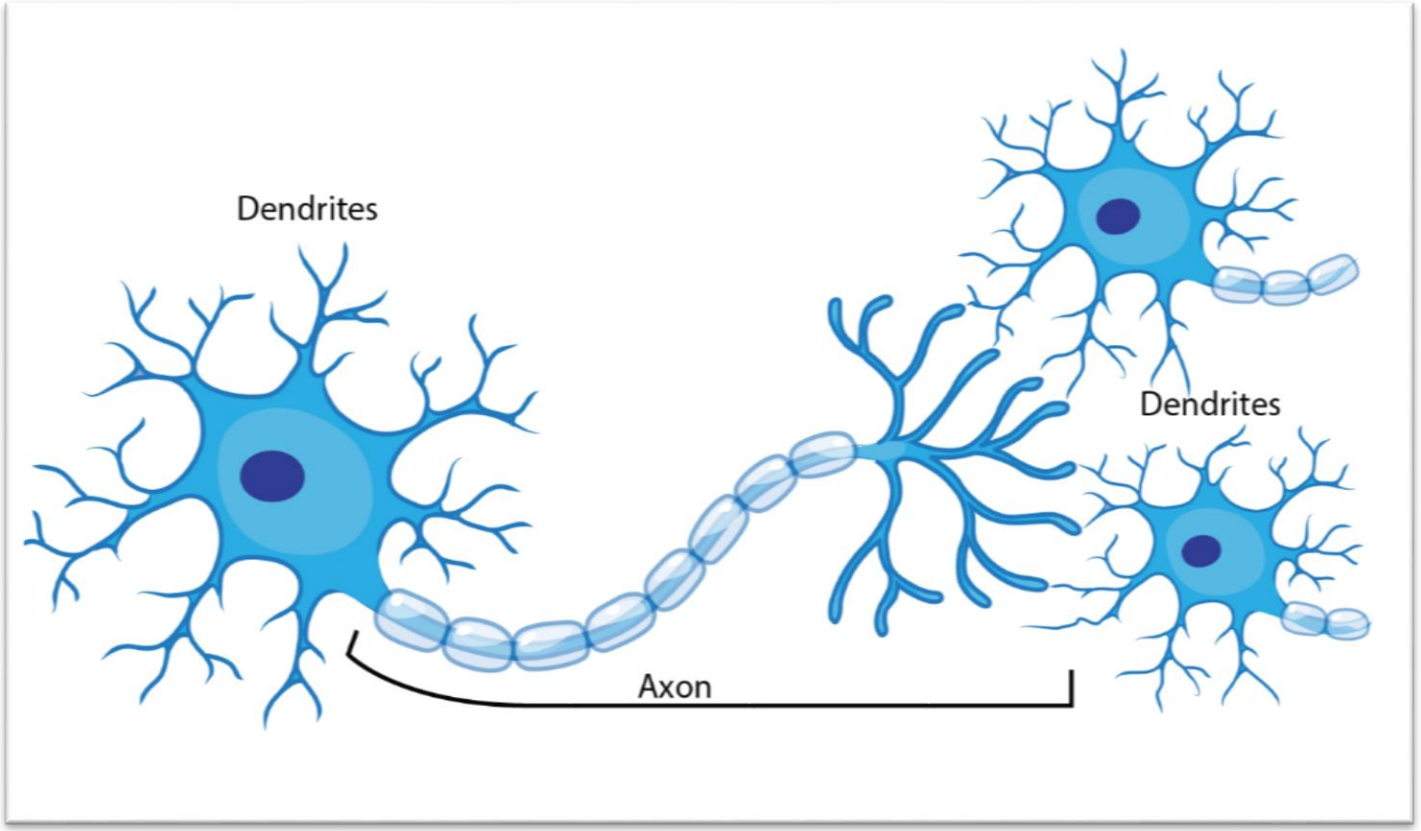
It's compatible

It's object-oriented

It has a lot of libraries

Deep Learning Magic competing with Natural Intelligence





Types of Learning Algorithms

- Supervised
  - Learns from examples which provide desired outputs for given inputs
- Unsupervised
  - Learns patterns in input data when no specific output values are given
- Reinforcement
  - Learns by an indication of correctness at end of some reasoning

Features

The features are the elements of your input vectors. The number of features is equal to the number of nodes in the input layer of the network

Category	Features
Housing Prices	No. of Rooms, House Area, Air Pollution, Distance from facilities, Economic Index city, Security Ranking etc.
Spam Detection	presence or absence of certain email headers, the email structure, the language, the frequency of specific terms, the grammatical correctness of the text etc.
Speech Recognition	noise ratios, length of sounds, relative power of sounds, filter matches
Cancer Detection	Clump thickness, Uniformity of cell size, Uniformity of cell shape, Marginal adhesion, Single epithelial cell size, Number of bare nuclei, Bland chromatin, Number of normal nuclei, Mitosis etc.
Cyber Attacks	IP address, Timings, Location, Type of communication, traffic details etc.
Video Recommendations	Text matches, Ranking of the video, Interest overlap, history of seen videos, browsing patterns etc.
Image Classification	Pixel values, Curves, Edges etc.

Weights

Weights correspond to each feature.

Weights denote how much the feature matters in the model.

Higher weight of a particular feature means that it is more important in deciding the outcome of the model.

Weights of a feature represent that how much evidence it gives in favor or against the current hypothesis in context of the existence or non-existence of the pattern you are trying to identify in the current input.

Generally weights are initialized randomly.

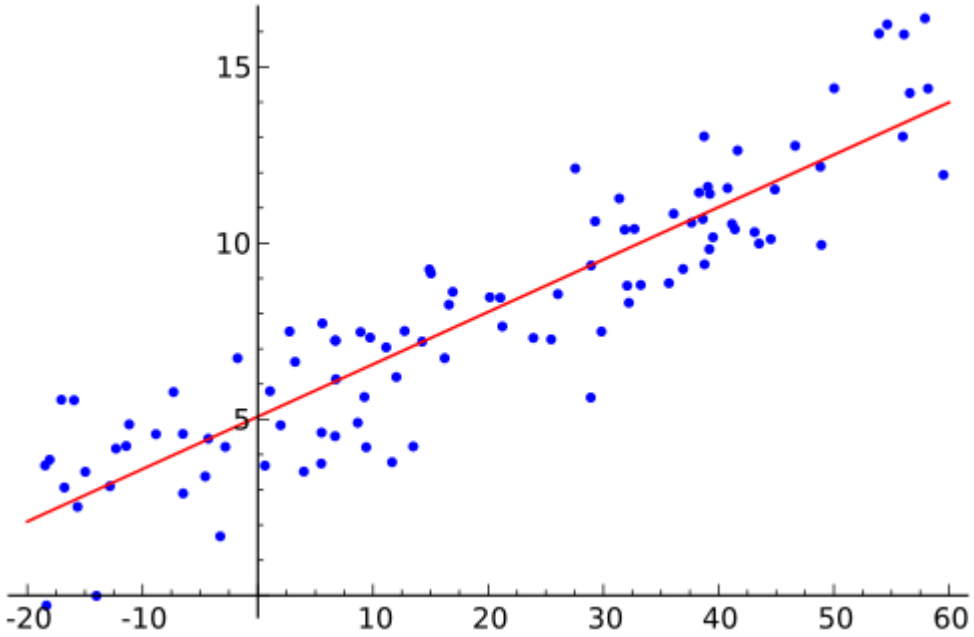
we try to bring them to near optimal values so that they are able to fit the model well and can help in prediction of unseen values

Linear Regression and Logistic Regression

Linear Regression: For applications where output will be a real value e.g. Predicting housing price, or predicting price of a share in stock market. In most cases we have multiple dependent variables, and we call it multiple linear regression

Logistic Regression: For applications where the output will be a binary value (0/1). E.g. whether this Medical Image depicts Tumor or not

Linear Regression



$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m}(\hat{Y}_i - Y_i)^2$$

Loss Function Squaring the Error difference

Benefits of squaring

Squaring always gives a positive value, so the sum will not be zero.

Squaring emphasizes larger differences—a feature that turns out to be both good and bad (think of the effect outliers have).

Linearity Vs. Non-Linearity

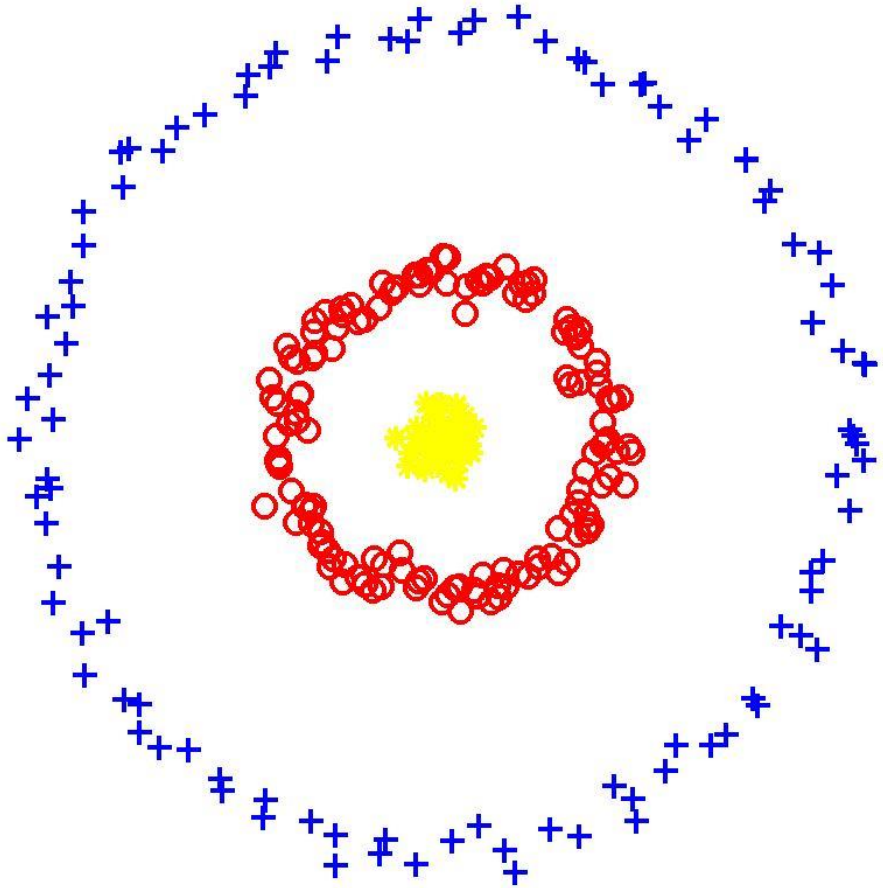
Simple multiplication of weights with inputs and giving the outputs will be linear function.

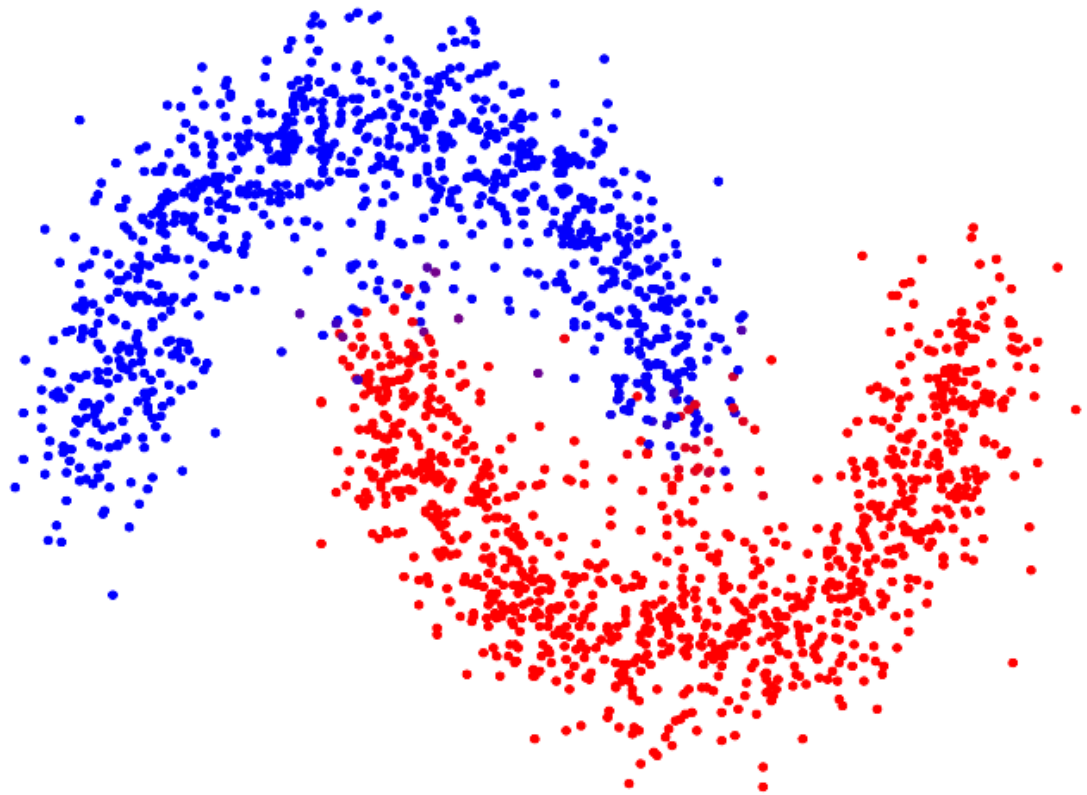
A linear function is just a polynomial of one degree and will not be able to learn complex functions.

Linear functions don’t have much expressive power and will loose out when solving complex problems.

Without activation functions (which will help us to achieve non-linearity) Neural Network will not be able to learn unstructured data like images, audio data, videos and text.

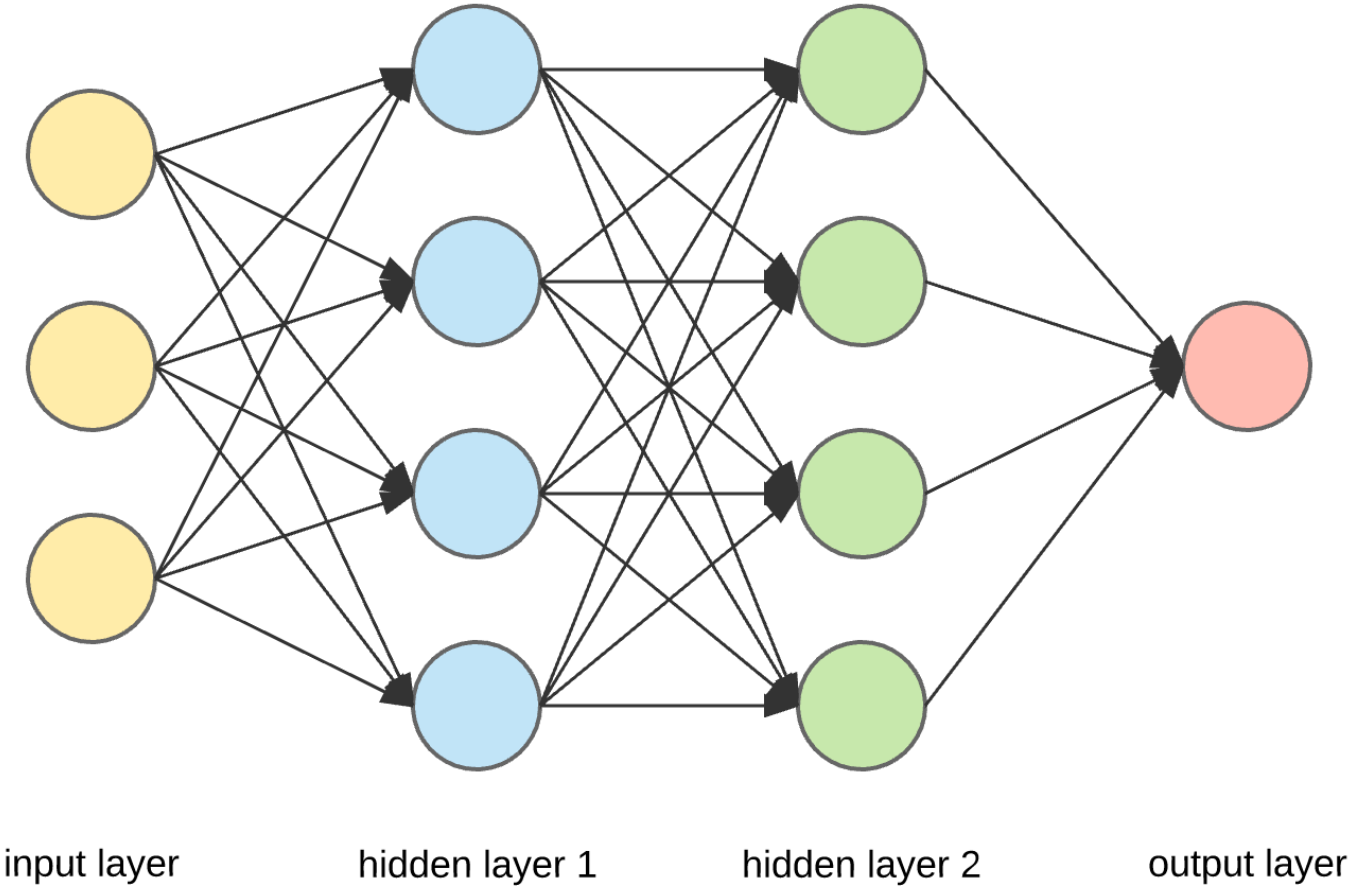
Non Linear Patterns



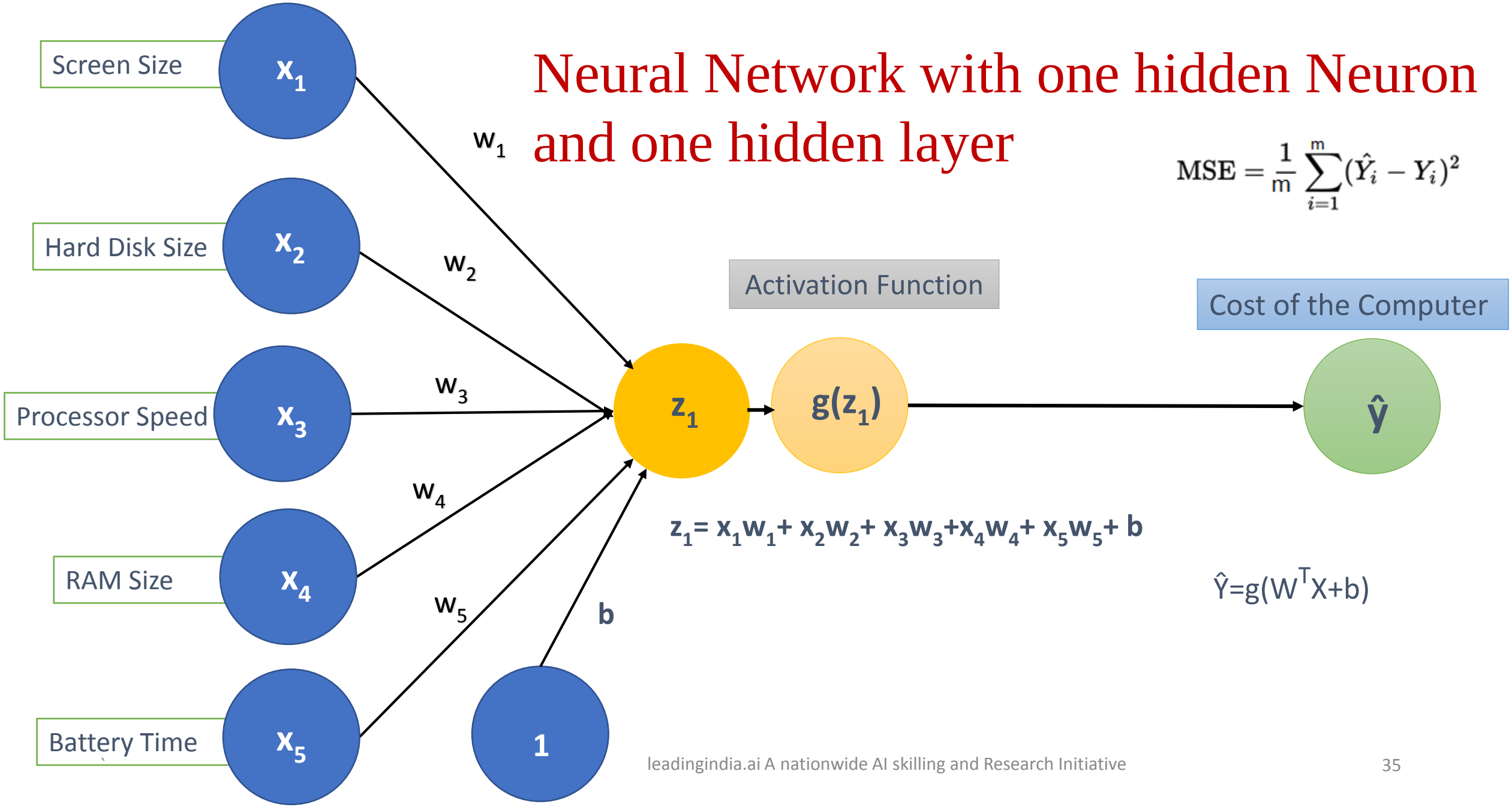


Logistic Regression

A Deep Neural Network

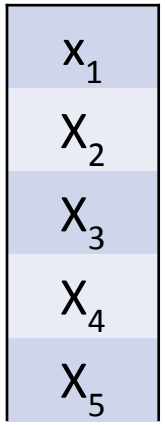


Neural Network with one hidden Neuron and one hidden layer



$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(\hat{Y}_i - Y_i)^2$$

Why $W^T X$

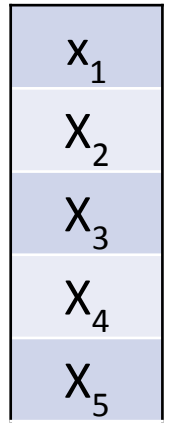


Matrix Size: (5X1)

Matrix Size: (5X1)

Multiplication of these two matrices is not possible

Matrix Size: (1X5)



Matrix Size: (5X1)

Activation Functions

Output from every neuron is generated after applying activation Function to the values being calculated with set of inputs and their weights.

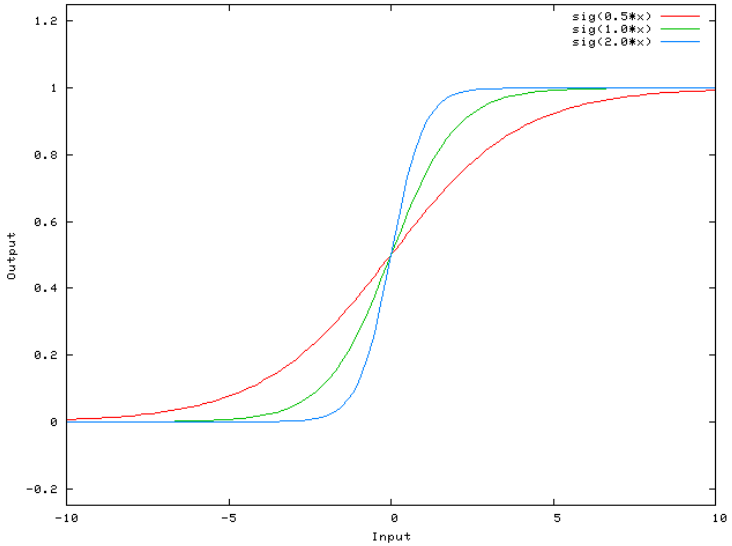
Most Popular Activation Functions are ReLU (Rectified Linear Units) and Sigmoid, Softmax and tanh

Activation functions should be differentiable. Non-linear Activation functions help you to bring non-linearity in the system

To transform/squash your input to a different space/domain and do some kind of thresholding

Bias

- The bias value allows the activation function to be shifted to the left or right, to better fit the data. Changes to the weights alter the steepness of curve, whilst the bias offsets it.
- Bias only influences the output values, it doesn't interact with the actual input data and are independent of the output from previous layers.
- Biases are tuned alongside weights by learning algorithms such as gradient descent.
- If all the inputs are zero for a particular layer, still with the bias term will have non zero output



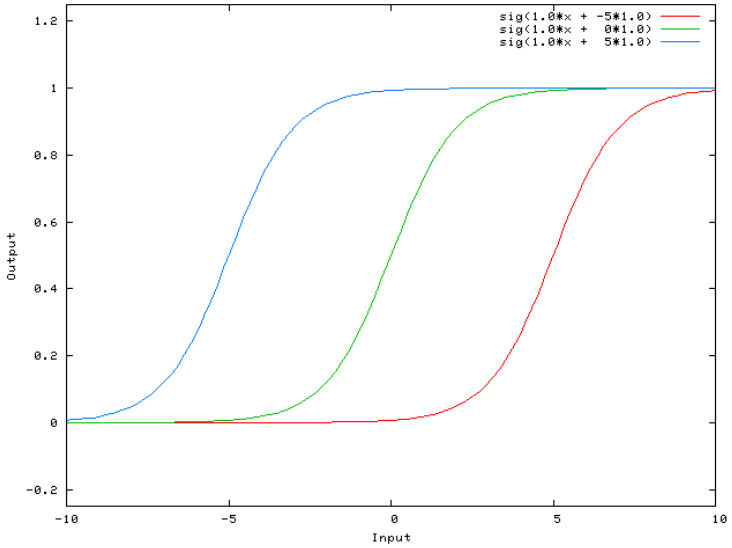


Image (8X8X3) pixels

		23	56	111	12	213	99	145	167
	37	121	14	74	224	121	14	74	224
41	51	100	17	69	33	100	17	69	33
67	121	189	19	36	12	189	19	36	12
99	142	158	223	14	45	158	223	14	45
88	153	93	117	21	67	93	117	21	67
33	167	62	119	43	89	62	119	43	89
22	189	19	21	65	09	19	21	65	09
12	21	06	47	01	24	189	111	66	
46	13	45	68	32	12	87	32		

192 Value Input Vector

23
56
:
:
65
09
37
:
:
66
41
:
:
12
87
32

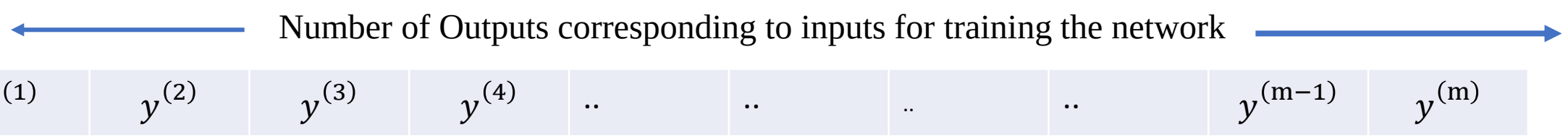
Shape of Your Input: **X** (n x m)

Number of IO pairs available for training the network

No of ***Data Points*** in Single Input

$x_1^{(1)}$	$x_1^{(2)}$	$x_1^{(3)}$	$x_1^{(4)}$	..	..	..	..	$x_1^{(m-1)}$	$x_1^{(m)}$
$x_2^{(1)}$	$x_2^{(2)}$	$x_2^{(3)}$	$x_2^{(4)}$	..	..	..	..	$x_2^{(m-1)}$	$x_2^{(m)}$
$x_3^{(1)}$	$x_3^{(2)}$	$x_3^{(3)}$	$x_3^{(4)}$	..	..	..	..	$x_3^{(m-1)}$	$x_3^{(m)}$
$x_4^{(1)}$	$x_4^{(2)}$	$x_4^{(3)}$	$x_4^{(4)}$	..	..	..	..	$x_4^{(m-1)}$	$x_4^{(m)}$
:	:	:	:	..	..	..	..	:	:
:	:	:	:	..	..	..	..	:	:
:	:	:	:	..	..	..	..	:	:
:	:	:	:	..	..	..	..	:	:
:	:	:	:	..	..	..	..	:	:
:	:	:	:	..	..	..	..	:	:
$x_{n-1}^{(1)}$	$x_{n-1}^{(2)}$	$x_{n-1}^{(3)}$	$x_{n-1}^{(4)}$	..	..	..	..	$x_{n-1}^{(m-1)}$	$x_{n-1}^{(m)}$
$x_n^{(1)}$	$x_n^{(2)}$	$x_n^{(3)}$	$x_n^{(4)}$	..	..	..	..	$x_n^{(m-1)}$	$x_n^{(m)}$

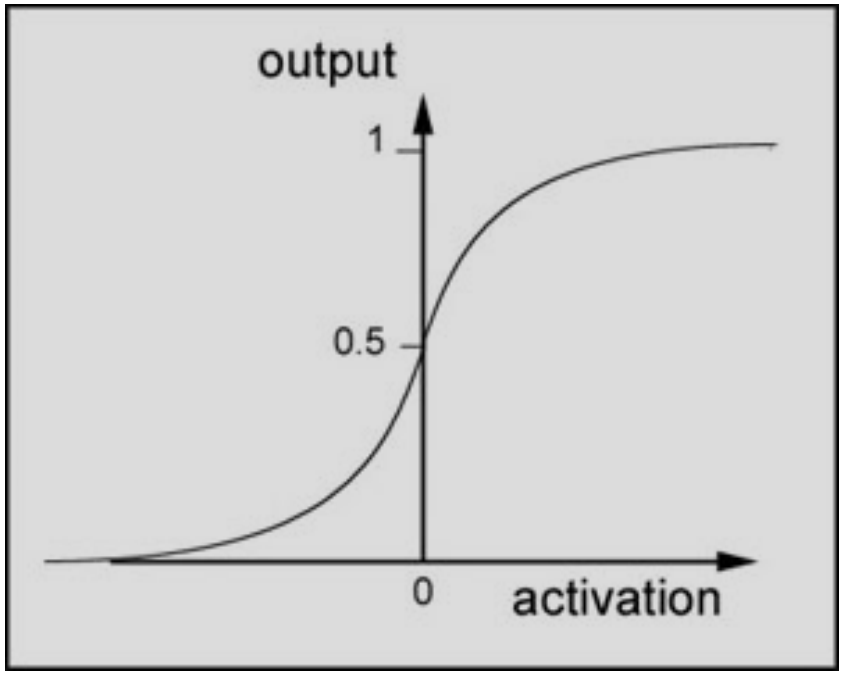
Shape of Your Output: Binary Classification



Y.Shape = (1 , m)

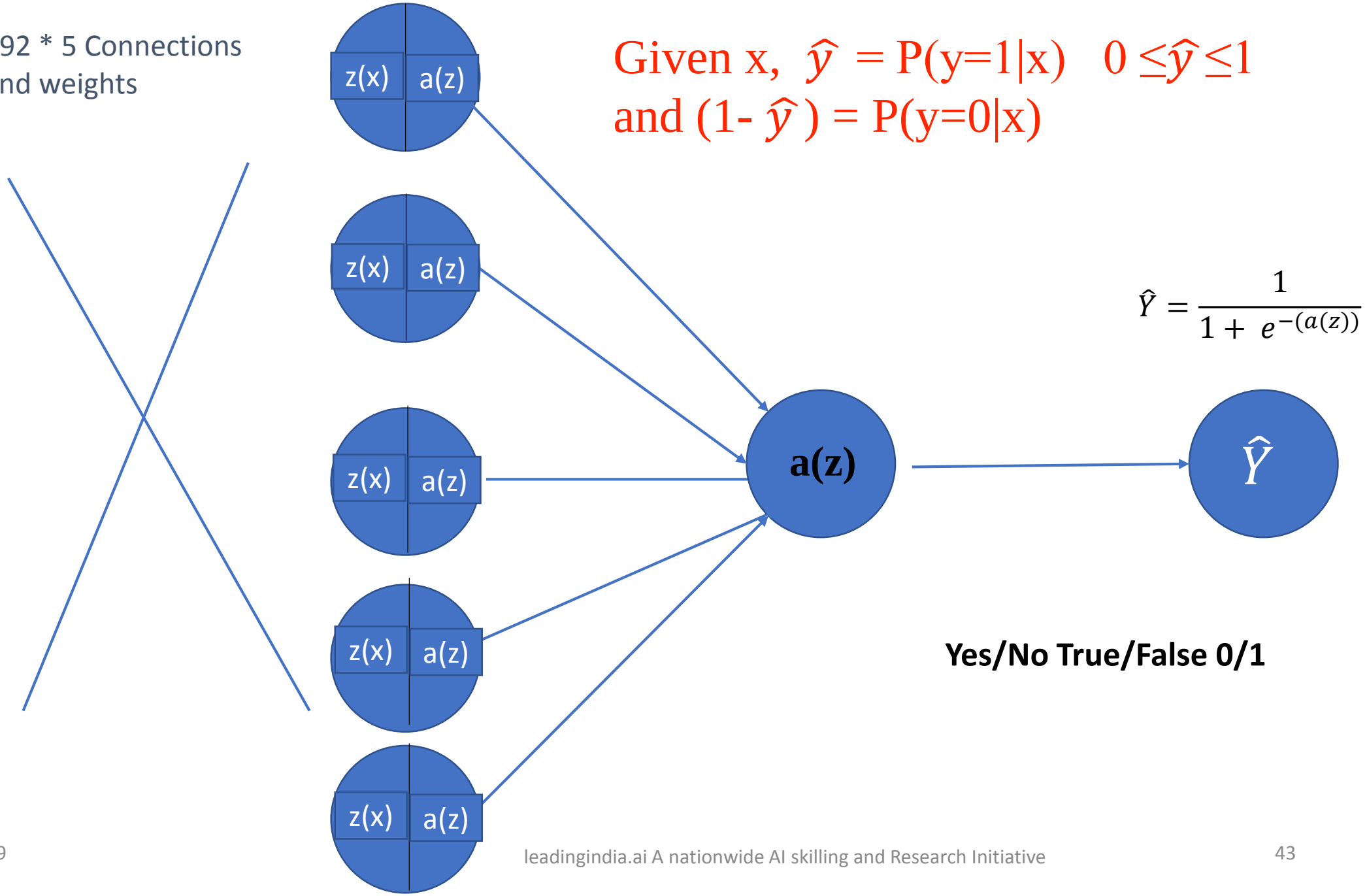
Sigmoid Activation Function

- $\hat{y}=\sigma\ (w^{T}x + b)$ where $\sigma(z)= \frac{1}{1+e^{-z}}$
- If z is very large then e^{-z} is close to zero and $\sigma(z) = \frac{1}{1+0} \approx 1$
- If z is very small then e^{-z} is large and $\sigma(z) = \frac{1}{1+Large\ Number} \approx 0$



23
56
:
:
65
09
37
:
:
66
41
:
:
12
87
32

192 * 5 Connections and weights



Given x, $\hat{y}$ = P(y=1|x) $0 \leq \hat{y} \leq 1$
and $(1-\hat{y})$ = P(y=0|x)

$$\hat{Y} = \frac{1}{1 + e^{-(a(z))}}$$

Yes/No True/False 0/1

Loss Function for logistic regression

L ($\hat{y}$, y) =-((ylog $\hat{y}$) + (1-y) (log (1- $\hat{y}$))

If y=1 L ($\hat{y}$, y)= -(log $\hat{y}$) Here, we want log $\hat{y}$ large , want $\hat{y}$ large that means $\hat{y} \approx 1$

If y=0 L($\hat{y}$, y)=-(log(1- $\hat{y}$) Here we want (log(1- $\hat{y}$) large, want $\hat{y}$ small means$\hat{y} \approx 0$

It helps us also to resolve the local minima problem and gives us a convex problem

Cost Function: Average of Loss across the whole input

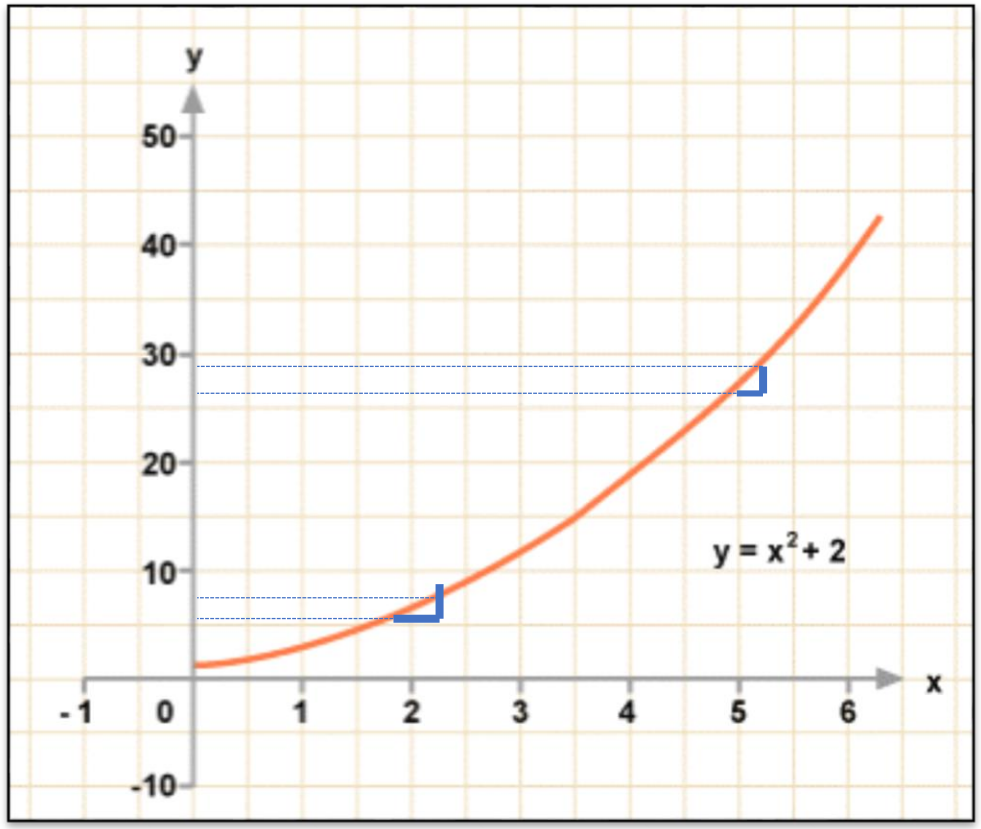
$\hat{y}^{(i)}=\sigma\,(w^{T}x^{(i)} + b)$ where $\sigma(z^{(i)})= {}^{1}/_{1+e^{-z^{(i)}}}$

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L\,(\hat{y}^{(i)}, y^{(i)})$$

$$= -\frac{1}{m}\sum_{i=1}^{m} (y^{(i)}\log \hat{y}^{(i)} + (1-y^{(i)})\log (1-\hat{y}^{(i)}))$$

Our objective is to find w,b that minimize J(w,b)

Derivatives



$y=f(x)=x^2+2$

$\frac{df(x)}{dx} = 2x$

x=2 f(x)=6

x=2.0001 f(x)=6.00040001

Slope at x=2 is .0004/.0001= 4

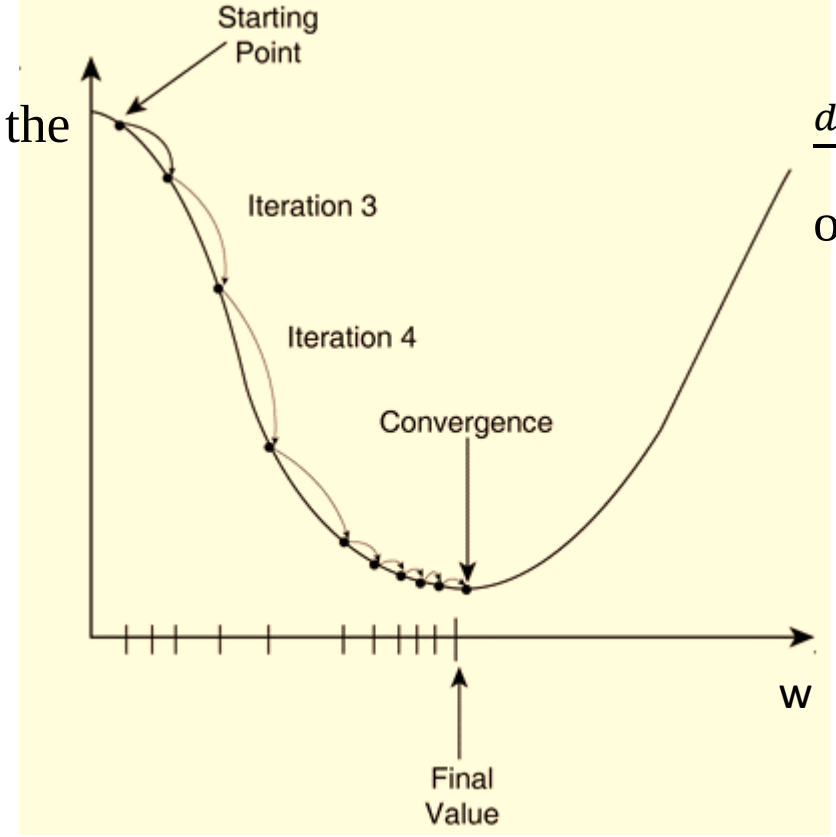
x=5 f(x) =27

x=5.0001 f(x) = 27.00100001

Slope at a =5 is .0010/.0001= 10

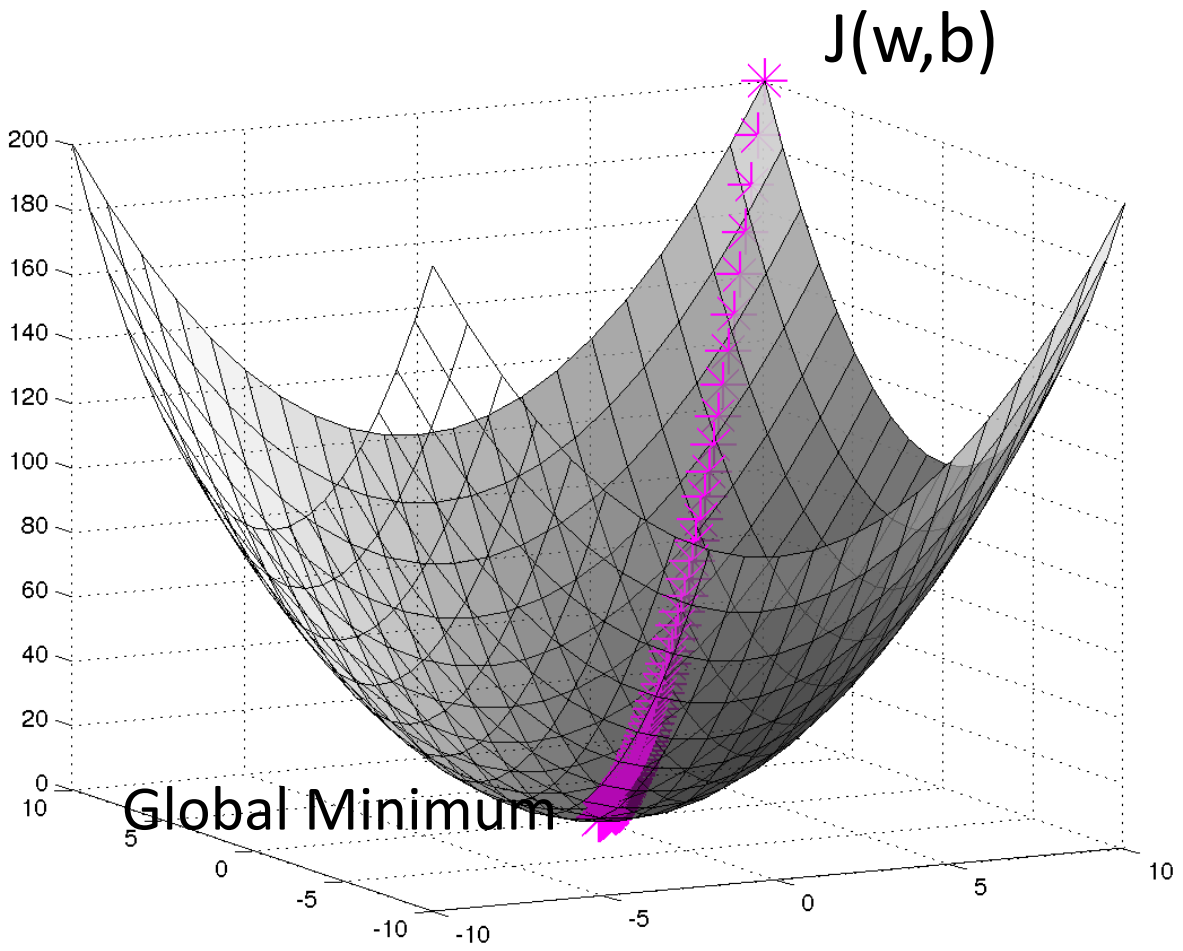
Gradient Descent single dimension

$\frac{dj(w)}{dw}$ is less than the optimum value



$\frac{dj(w)}{dw}$ is more than the optimum value

Gradient Descent



How the weights will change

w= w-α $\frac{dJ(w,b)}{dw}$ (Mathematicians write it as w= w-α $\frac{\partial(w,b)}{\partial w}$) Python w= w-α dw

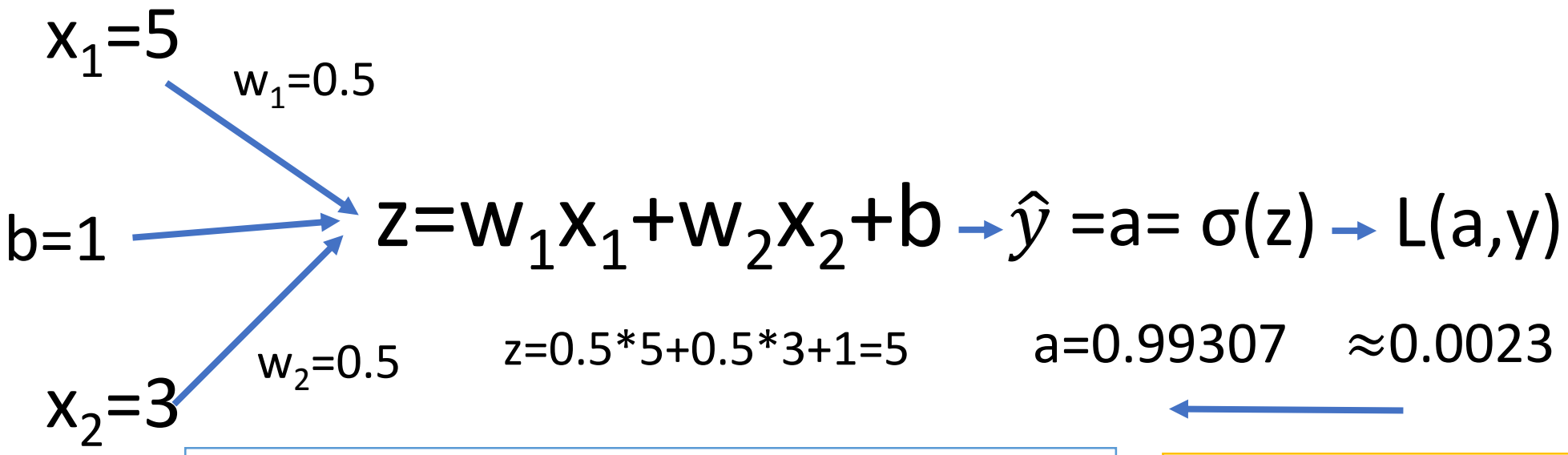
b= b-α $\frac{dJ(w,b)}{db}$ (Mathematicians write it as b= b-α $\frac{\partial(w,b)}{\partial b}$) Python b= b-α db

This process will repeat until we find or reach near the global minimum.

α is the learning rate which will decide that how fast or slow we are going towards the global minima. How big or small steps we want to take. Both have their own limitations and advantages.

Derivatives in logistic regression

Original Value of y=1



db=dz

$dz=\frac{dL(a,y)}{dz}=\frac{dL}{da}\frac{da}{dz}=(\frac{-y}{a}+\frac{1-y}{1-a})$*a(1-a)=a-y=-0.00693

$da=\frac{dL(a,y)}{da}=\frac{-y}{a}+\frac{1-y}{1-a}$ = -1.006978

α= 0.01

$dw_1=\frac{dL(a,y)}{dw_1}$ =x_1.dz=5*-0.00695=-0.03475

$dw_2=\frac{dL(a,y)}{dw_2}$ =x2.dz=3*-0.00695=-0.02085

$w_1=w_1-\alpha dw_1$ = 0.5-(0.01*-0.03475)=0.503475

$w_2=w_2-\alpha dw_2$ = 0.5-(0.1*-0.02085)=0.502085

$b=b-\alpha db$ =1-(0.1)*(-0.00693)=1.000693

Logistic Regression on whole training set

J=0, dw_i=0,db=0

for i=1 to m

$z^{(i)}=w^T x^{(i)}+b$

$a^{(i)}=\sigma z^{(i)}$

$J+=-[y^{(i)}\log a^{(i)} + (1-y^{(i)}) \log (1- a^{(i)})]$

$dz^{(i)}=a^{(i)}-y^{(i)}$

for k=1 to n #no of features

$dw_k+=x_k^{(i)}dz^{(i)}$

db+=$dz^{(i)}$

J/=m

db/=m

for i=1 to n

dw_i/=m

$w_i=w_i-\alpha\ dwi$

b=b-αdb

Vectorized

for iteration (epoch) in range (1000)

$Z=w^TX+b$ (In python np.dot(w.T,X)+b)

$A=\sigma Z$

$dZ=A-Y$

$dw=\frac{1}{m}XdZ^T$

$db=\frac{1}{m}$ np.sum(dz)

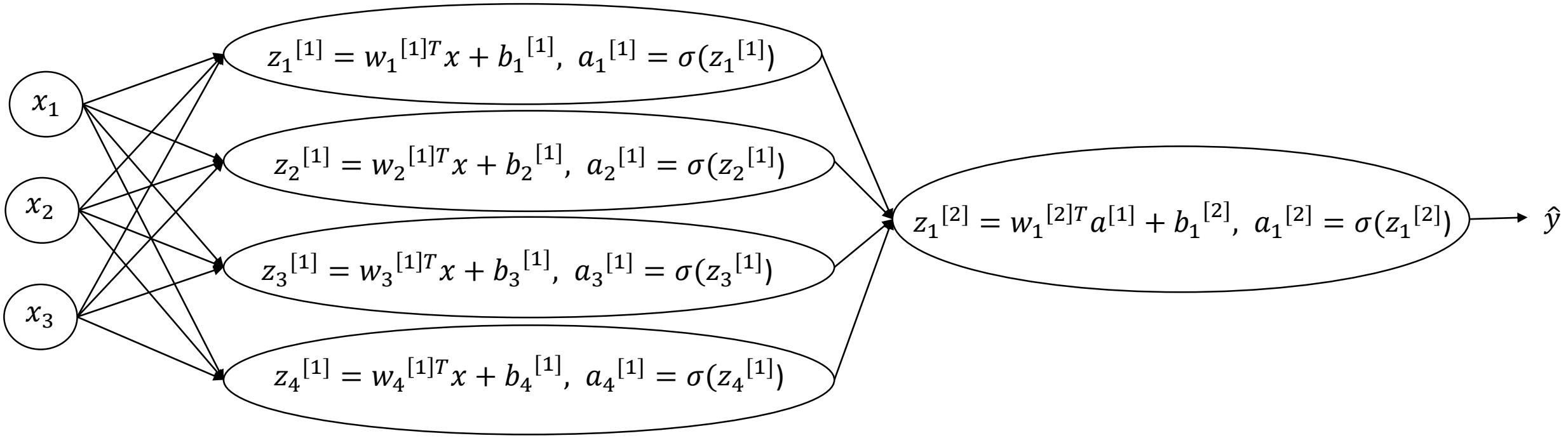
$w=w-\alpha\ dw$

$b=b-\alpha db$

Neural Network Representation

2 layer neural network

$a_i^{\,l}$ $a^{layer}_{node\ in\ layer}$



$a^{[0]}$ Layer 0
Input Layer

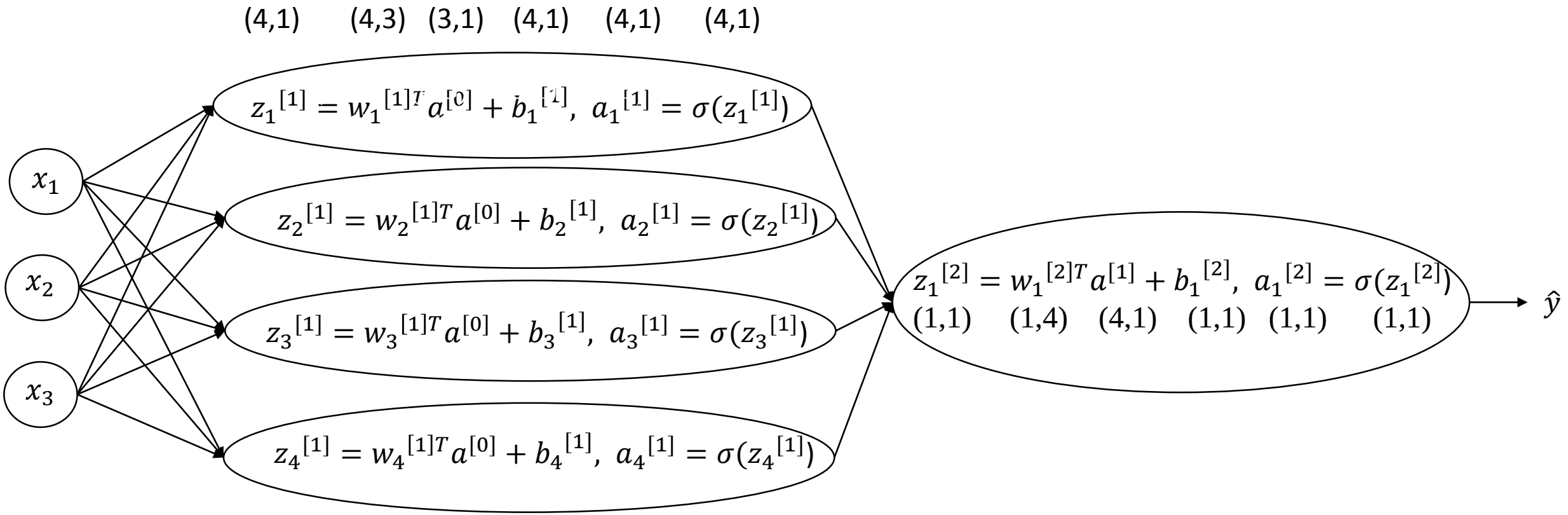
$a^{[1]}$ Layer 1 $w^{[1]}$ $b^{[1]}$ $z^{[1]}$
Hidden Layer

$a^{[2]}$ Layer 2 $w^{[2]}$ $b^{[2]}$ $z^{[2]}$
Output Layer

Neural Network Representation

2 layer neural network

$a_i^{\,l}$ $a^{layer}_{node\ in\ layer}$



$a^{[0]}$ Layer 0
Input Layer

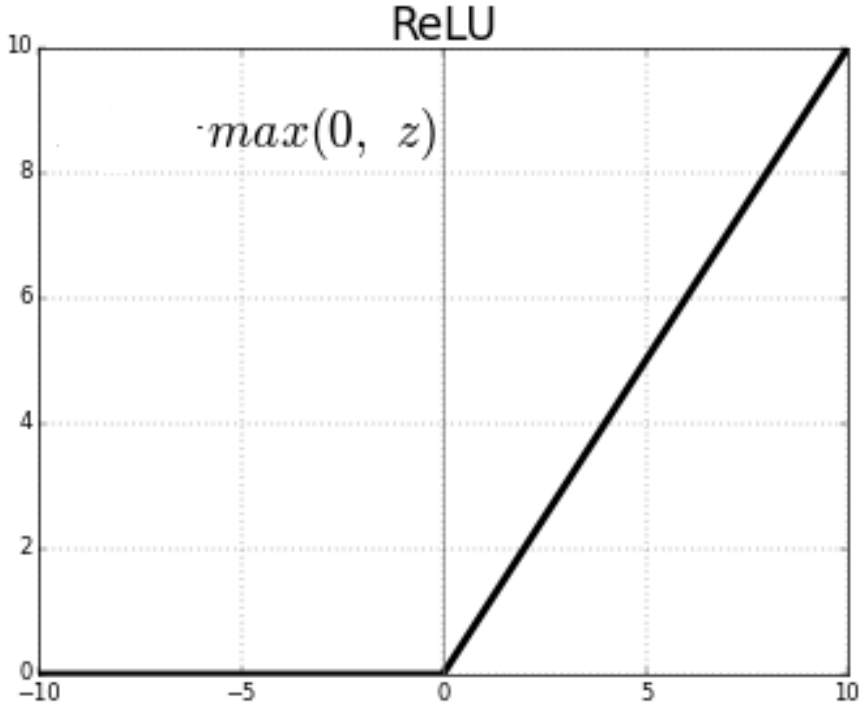
$a^{[1]}$ Layer 1 $w^{[1]}$ $b^{[1]}$ $z^{[1]}$
Hidden Layer

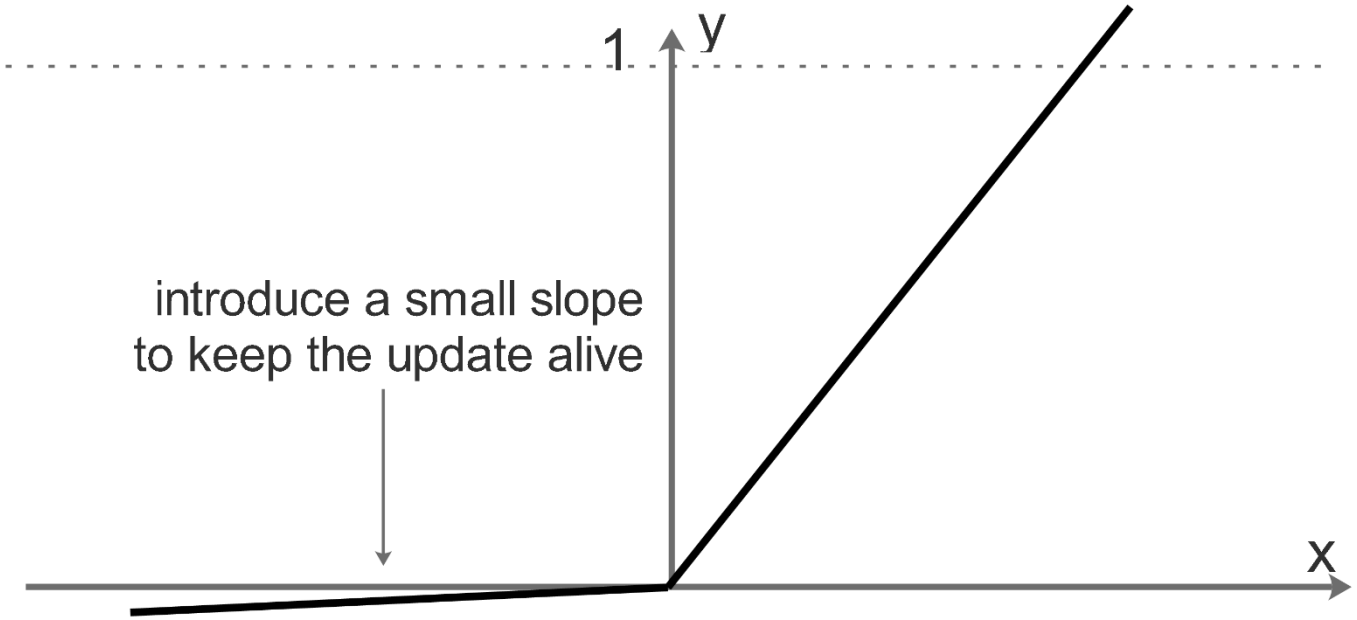
$a^{[2]}$ Layer 2 $w^{[2]}$ $b^{[2]}$ $z^{[2]}$
Output Layer

Weight Matrix

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}
w_{41}	w_{42}	w_{43}

ReLU and Leaky ReLU





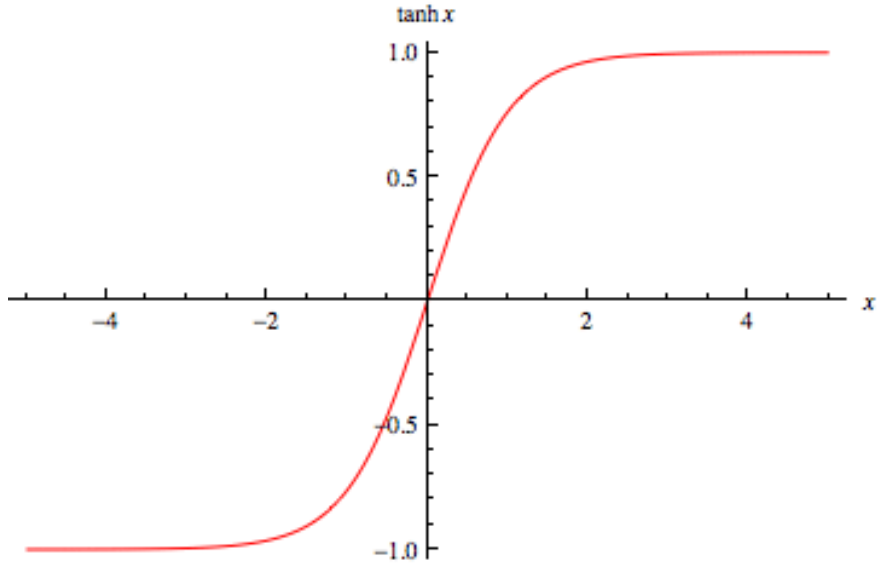
Leaky ReLU max(0.1z,z)

Rectified Linear Unit is the default activation function being used now. If you see the left part of the function, you can see that it is not zero, but almost zero. To resolve the issue of dead neurons people use Leaky ReLU

Activation Functions: tanh

Tanh activation function is more preferred than the sigmoid function.
It is the shifted version of sigmoid function with a mean value of zero.
It has better centering effect for the activation function to be used on the hidden layer.
For binary classification problem at the output layer we use the Sigmoid function.

$$\tanh = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$



Problem in both Sigmoid and tanh is that the slope of the curve except the middle region is too small and goes close to zero. This create a serious issue with gradient descent and learning becomes very slow.

Softmax Function

When we have to classify in multiple categories then softmax function is useful. For example if you want to categorize pictures into

A) scene b) Animals c) Humans d) Vehicles then in that case we will have four outputs from the softmax function which will give us the probabilities of each of these categories.

Sum of the probabilities will be one and that with the highest probability will be shown as the answer.

Understanding softmax

$$z^{[L]} = \begin{pmatrix} 5 \\ 2 \\ -1 \\ 3 \end{pmatrix} \qquad t = \begin{pmatrix} e^{5} \\ e^{2} \\ e^{-1} \\ e^{3} \end{pmatrix}$$

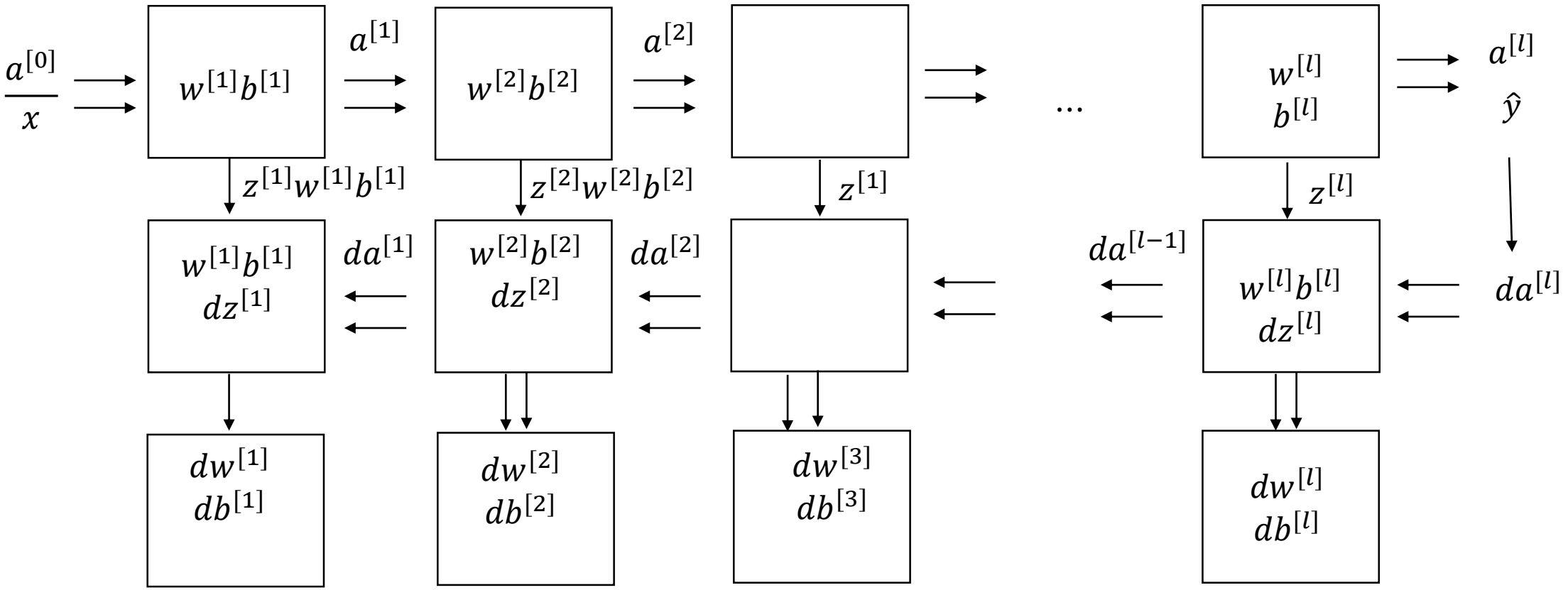
$$a^{[l]} = g^{[L]}\left(z^{[L]}\right) = \begin{pmatrix} e^{5}/(e^{5}+e^{2}+e^{-1}+e^{3}) \\ e^{2}/(e^{5}+e^{2}+e^{-1}+e^{3}) \\ e^{-1}/(e^{5}+e^{2}+e^{-1}+e^{3}) \\ e^{3}/(e^{5}+e^{2}+e^{-1}+e^{3}) \end{pmatrix} = \begin{pmatrix} 0.842 \\ 0.042 \\ 0.002 \\ 0.114 \end{pmatrix}$$

"Hard Max"

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

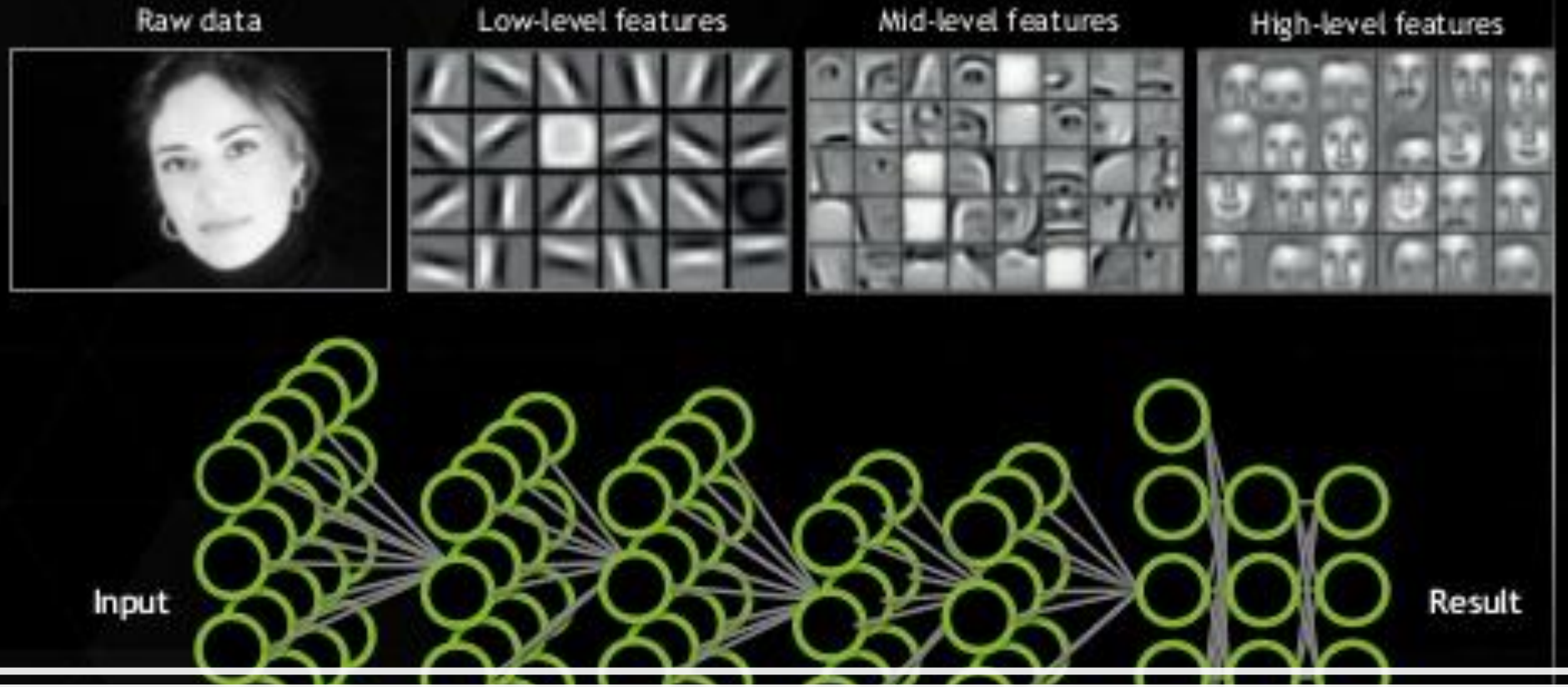
Softmax regression generalizes logistic regression to C classes. If c=2, softmax reduces to logistic regression.

Forward and Backward Functions



$$w^{[l]} = w^{[l]} - \propto dw^{[l]}$$
$$b^{[l]} = b^{[l]} - \propto db^{[l]}$$

DEEP NEURAL NETWORK (DNN)



Application components:

Task objective
e.g. Identify face
Training data
10-100M images
Network architecture
~10 layers
1B parameters
Learning algorithm
~30 Exaflops
~30 GPU-days

What Happens Inside a Network