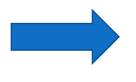# Human Brain VS Computer

Motivation

Human mind   - Good at image recognition, pattern recognition etc
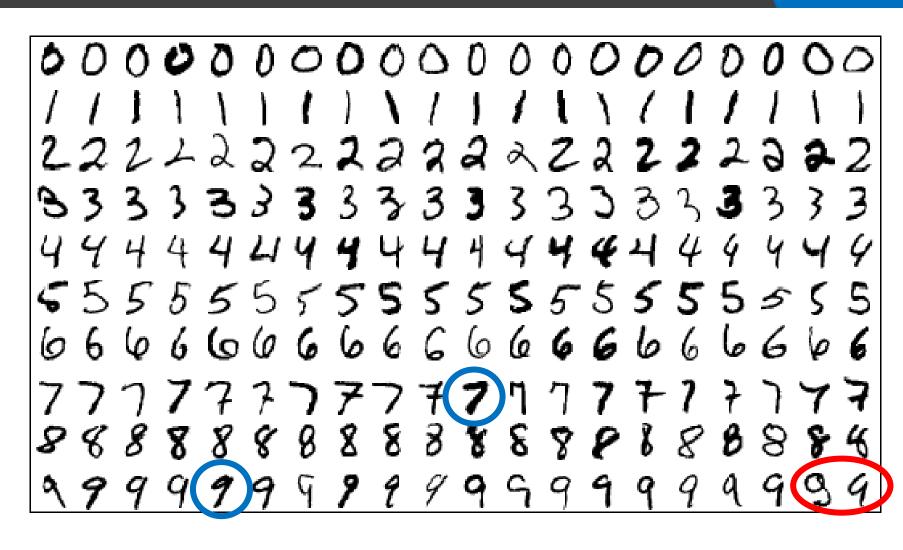Computer   - Good at arithmetic calculations

$$2574304 \times e^{354} \div \tan 5.1\pi$$

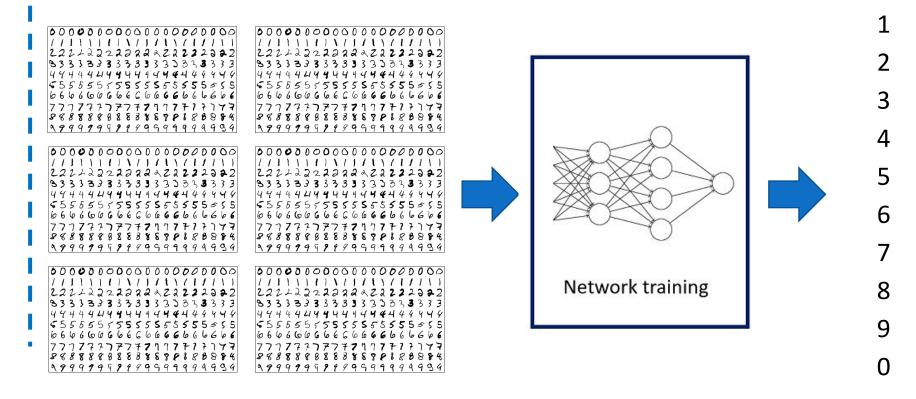# Handwriting recognition

Making precise rules is difficult

Neural Networks

Neural Networks creates own complex pattern recognition rules

Pattern recognition

Training data

Network training

Future Prediction

1
2
3
4
5
6
7
8
9
0

Dataset

Fashion MNIST
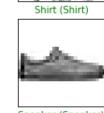
We will classify images into 10 fashion items
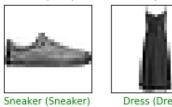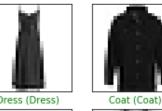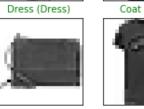
Course Flow

Course Flow

Single cell - Perceptron

Multi level perceptron

Forward and Backward propagation

Stochastic Gradient descent

Implementation in Python

Perceptron
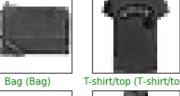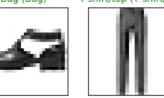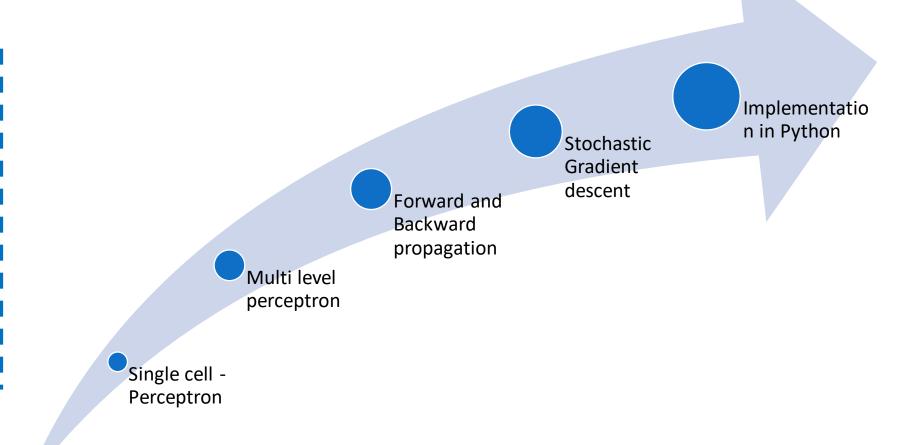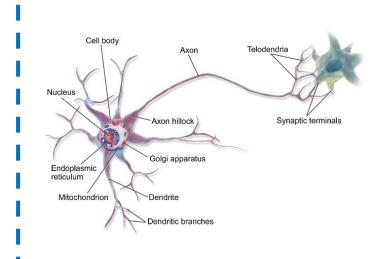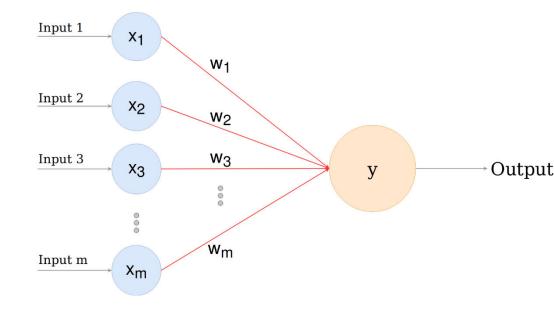
Artificial Neuron
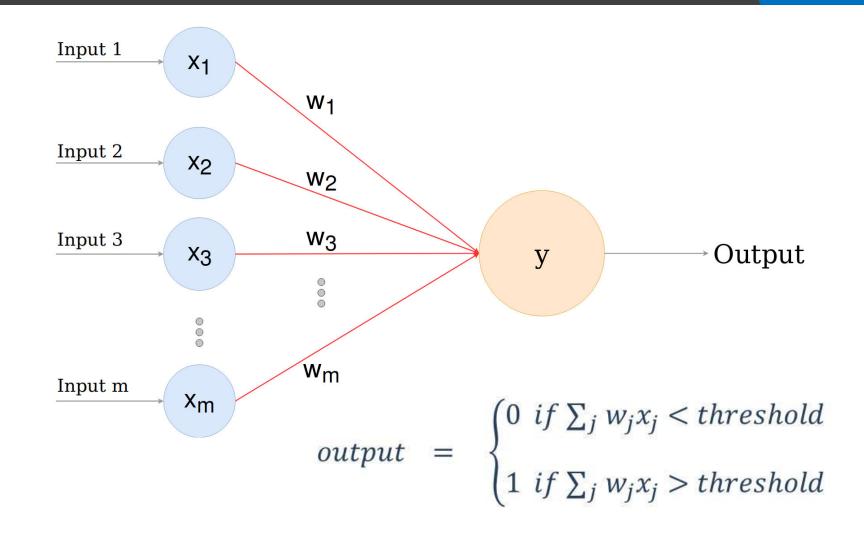
Biological Neuron

Artificial Neuron

Perceptron
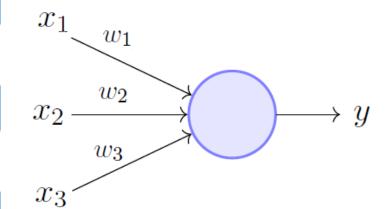
Artificial Neuron

$$output = \begin{cases} 0 \; if \; \sum_j w_j x_j < threshold \\ 1 \; if \; \sum_j w_j x_j > threshold \end{cases}$$

Example

Purchasing a Shirt

Color
- Blue or Not

Sleeves
- Full or half

Fabric
- Cotton or not

x_1 w_1

x_2 w_2

w_3

x_3

y

You will buy the shirt or not

Purchasing a Shirt

Color
- Blue or Not

Sleeves
- Full or half

Fabric
- Cotton or not

x_1 **7**

x_2 **4**

x_3 **2**

Threshold Value

8 y

You will buy the shirt or not

Example

Purchasing a Shirt

Color
- Blue or Not

Sleeves
- Full or half

Fabric
- Cotton or not

x_1 [7]

x_2 [4]

x_3 [2]

Threshold Value

[8] y
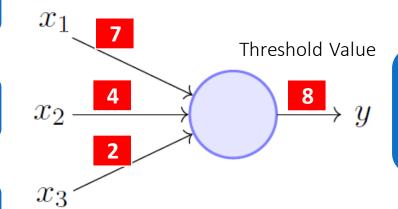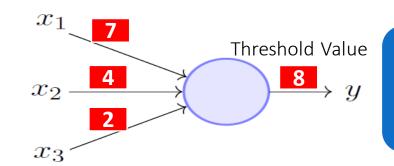
You will buy the shirt or not

Color	Sleeves	Fabric	Calculated Sum	Threshold	Buy / Not Buy
Blue	Half	Non Cotton	7*1 + 4*0 + 2*0 = 7	8	Not buy
Blue	Full	Non Cotton	11	8	Buy
Not Blue	Full	Cotton	6	8	Not Buy

Purchasing a Shirt

Color
- Blue or Not

Sleeves
- Full or half

Fabric
- Cotton or not

x_1 **8**

x_2 **6**

x_3 **3**

Threshold Value

16

y

You will buy the shirt or not

Perceptron
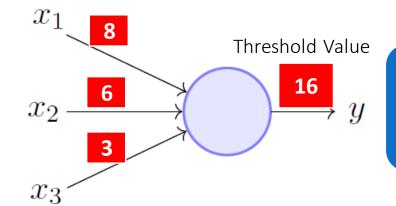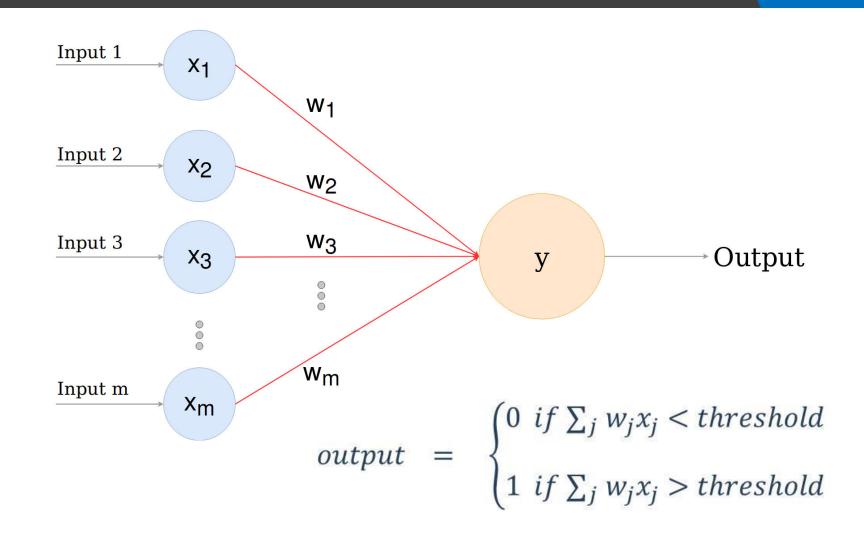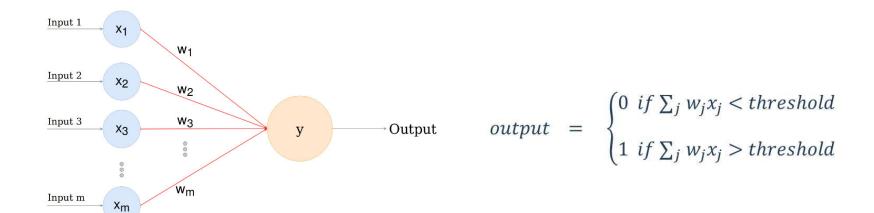
Removing Binary Restriction

$$output = \begin{cases} 0 \ if \ \sum_j w_j x_j < threshold \\ 1 \ if \ \sum_j w_j x_j > threshold \end{cases}$$

$$output = \begin{cases} 0 \ if \ \sum_j w_j x_j < threshold \\ 1 \ if \ \sum_j w_j x_j > threshold \end{cases}$$

Standard Equation
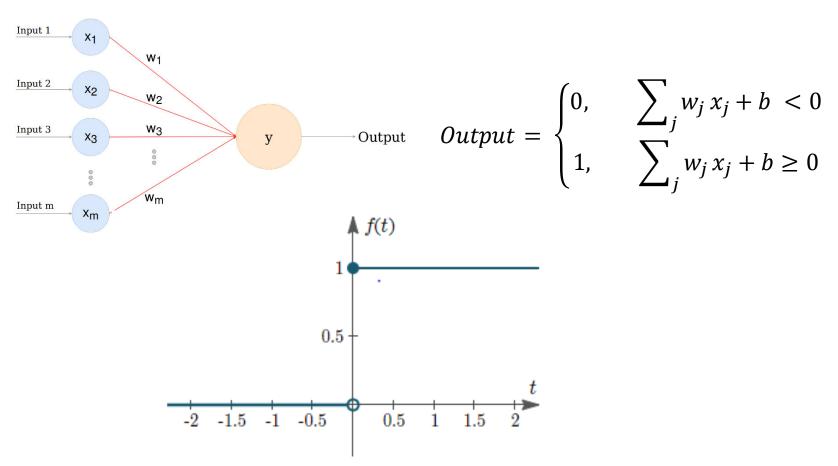
$$Output = \begin{cases} 0, & \sum_j w_j x_j + b < 0 \\ 1, & \sum_j w_j x_j + b \geq 0 \end{cases}$$

b is called Bias

Perceptron
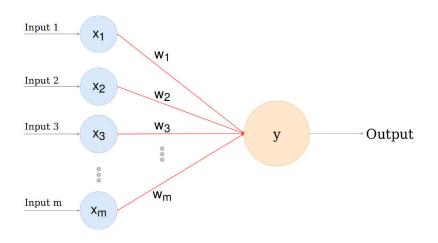
Graphical Representation

$$Output = \begin{cases} 0, & \sum_j w_j\, x_j + b < 0 \\ 1, & \sum_j w_j\, x_j + b \geq 0 \end{cases}$$

Step Activation function

Perceptron

Sigmoid Activation

$$Output = \begin{cases} 0, & \sum_j w_j x_j + b < 0 \\ 1, & \sum_j w_j x_j + b \geq 0 \end{cases}$$

Step Activation function

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

Sigmoid Activation function

Perceptron

$$\phi(z) = \frac{1}{1 + e^{-z}}$$
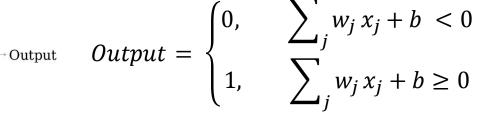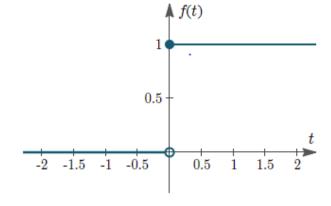
Sigmoid Activation function
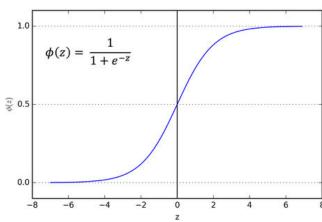
Sigmoid Activation

- Sigmoid is better because it is less sensitive to individual observation

- Artificial neuron with sigmoid activation is called sigmoid or logistic neuron

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

$$Output = \frac{1}{1 + \exp(-\sum_j w_j x_j - b)}.$$

Making Networks
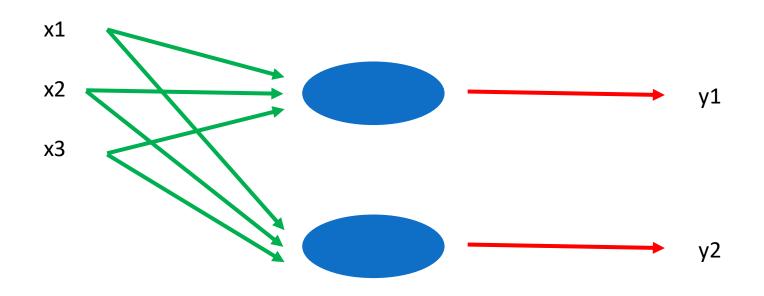
Two types of Stacking

Parallel

Sequential

Making Networks
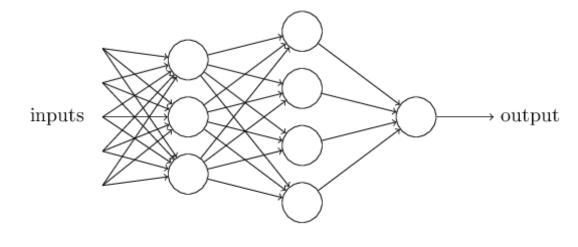
Parallel Stacking
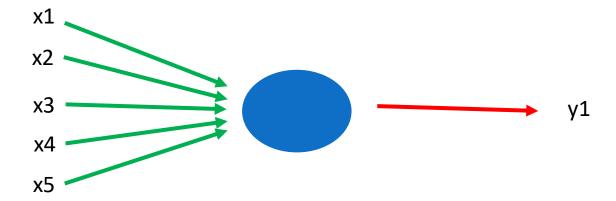
With parallel stacking we can get multiple outputs with the same input

Sequential Stacking
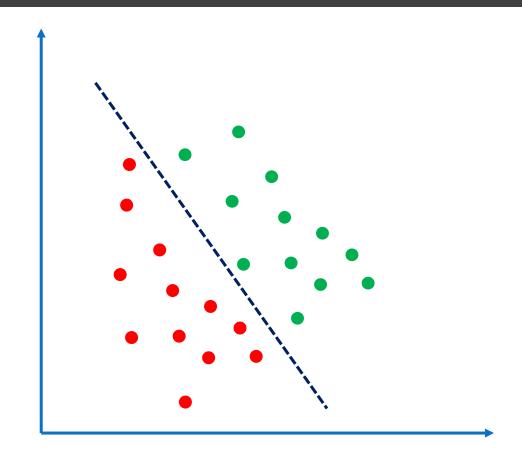
Why not use a single neuron

x1
x2
x3
x4
x5

y1

Making Networks
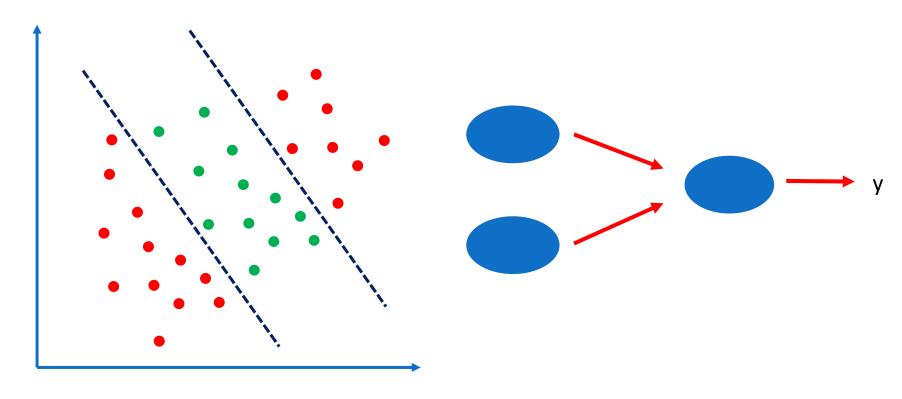
Sequential Stacking

Single neuron can handle such linear classification problem
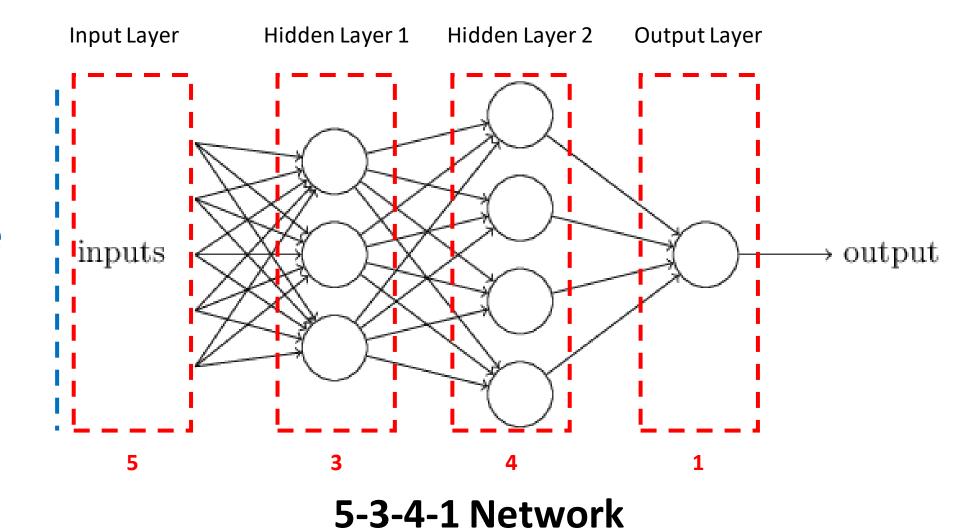
Sequential Stacking

Each neuron can focus on the particular features of the object instead of the final outcome

Making Networks

Nomenclature

Input Layer   Hidden Layer 1   Hidden Layer 2   Output Layer

inputs

output

5   3   4   1

5-3-4-1 Network

Nomenclature

inputs ... output
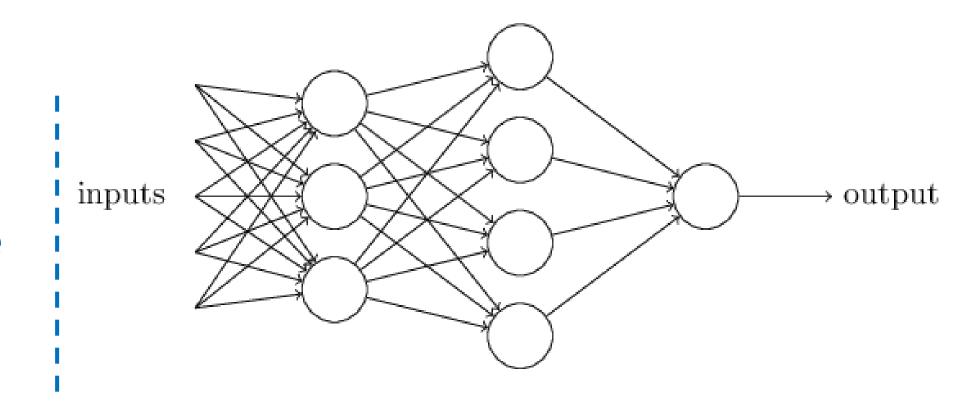
Feed Forward Network — One directional processing
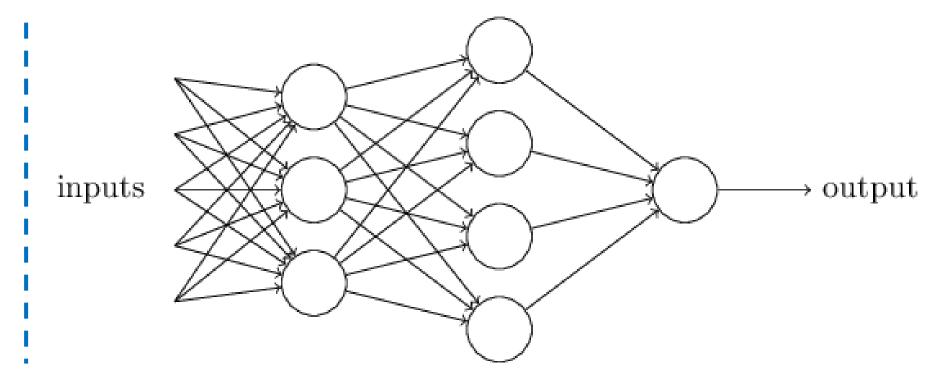
Fully connected network — Output from a neuron goes to all neurons of next layer

Deep Learning

Such artificial neural networks primarily constitutes deep learning

Deep Learning

inputs → output

More number of layers =>   Deeper network =>   More complex relationships

Neural Network

How it works

Covered till Now

- What is a neural network
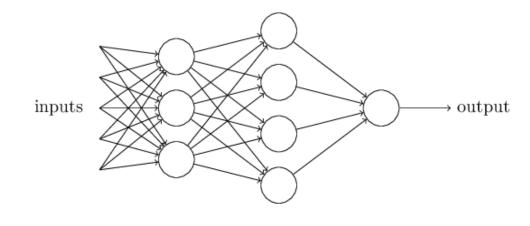
Now we are going to learn

- How does a neural network works

Problem Statement

Quick Recap

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

$$Output = \frac{1}{1+\exp(-\sum_j w_j x_j - b)}.$$
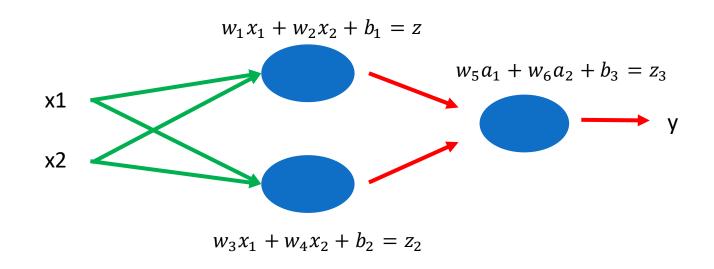
Problem Statement

- Establish the values of weights and biases so that predicted output is as close to actual output as possible

Example

$$w_1 x_1 + w_2 x_2 + b_1 = z$$

$$w_5 a_1 + w_6 a_2 + b_3 = z_3$$

x1

x2

y

$$w_3 x_1 + w_4 x_2 + b_2 = z_2$$
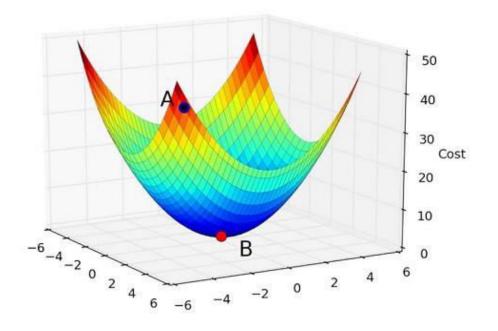
Variables to be established in this neural network

- Weights   - W1, W2……….W6

- Biases   - B1, B2, B3

Total   - 9 variables

Gradient Descent

- GD is an optimization technique to find minimum of a function
- Better than other technique such as OLS when we have large number of features and complex relationships

Gradient Descent

Process

Step 1
- Assign random W and B values

Step 2
- Calculate final output using these values

Step 3
- Estimate error using error function

Step 4
- Find those W and B which can reduce this error

Step 5
- Update W and B and repeat from step 2

Initialization

Forward Propagation

Backward Propagation
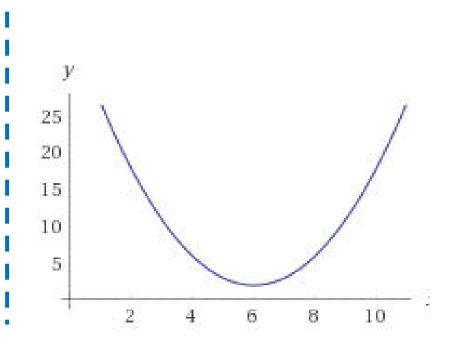
Implementation of GD

Gradient Descent

Gradient Descent
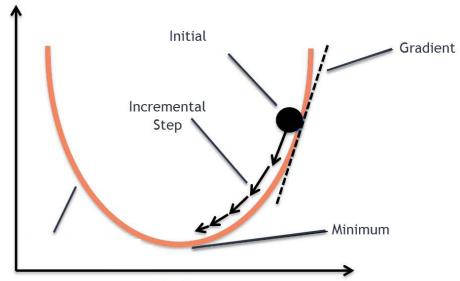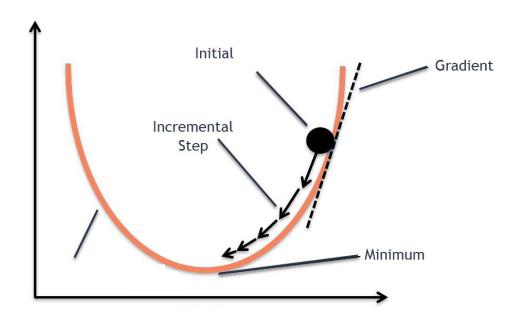
1. Start at a random point

2. Find out the **instantaneous slope** at that point

3. Slightly **move** in the direction of **steepest slope**

4. Reiterate
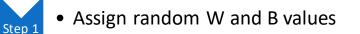
Gradient

Descent

Gradient Descent

Gradient Descent

Step 1	• Assign random W and B values
Step 2	• Calculate final output using these values
Step 3	• Estimate error using error function
Step 4	• Find those W and B which can reduce this error
Step 5	• Update W and B and repeat from step 2

Error Function

Error Function

Assume predicted output = 0.3 , actual output = 0

Distance $= 0 - 0.3 = -0.3$

Error Function $_1$ $= |-0.3| = 0.3$

Error Function $_2$ $= (-0.3)^2 = 0.09$
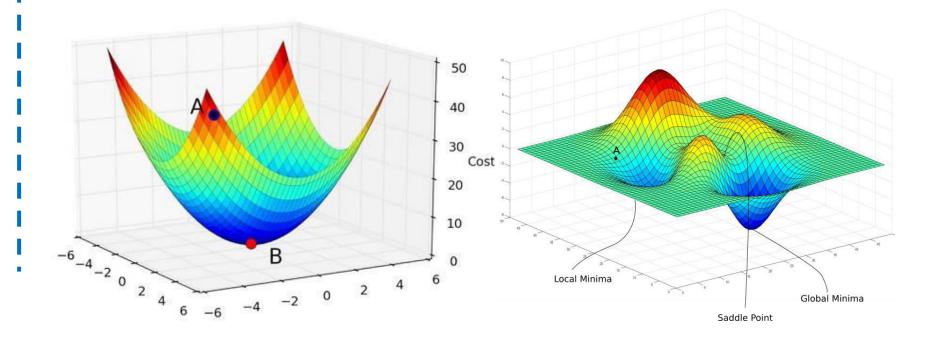
Square function works well with regression but not with classification

Gradient Descent

Cross Entropy Error Function

$$= \quad -y \log(y') - (1-y) \log(1-y')$$

Error Function

Gradient Descent

Error Function

Cross Entropy Error Function

$$= \quad -y \log(y') - (1 - y) \log(1 - y')$$

Assume actual output = y = 1 ,

Error   = - [1(log(y')) + (1-1)(log(1-y'))]

Error   = - [log(y')]

To minimize error, we have to minimize –log(y')

i.e. maximize log(y')

$\Rightarrow$ Maximize y'

Since y' lies between 0 and 1, y' should be as close to 1 as possible
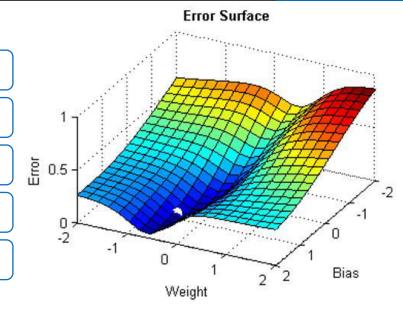
Gradient Descent

Back Propagation

Step 1	• Assign random W and B values
Step 2	• Calculate final output using these values
Step 3	• Estimate error using error function
Step 4	• Find those W and B which can reduce this error
Step 5	• Update W and B and repeat from step 2

Error Surface

$$w = w - \alpha\Delta w$$

$$b = b - \alpha\Delta b$$

α is learning rate, $\Delta w \ and \ \Delta b$ are unit steps

Alpha determines number of steps we take in downward direction

Gradient Descent

Back Propagation

$$w = w - \alpha \Delta w$$

$$b = b - \alpha \Delta b$$

To find $\Delta w \; and \; \Delta b$

We do back propagation

Example

x1

x2

Y'

$$w_1 x_1 + w_2 x_2 + b_1 = z$$

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

Error Function

$$= \quad -y \log(y') - (1 - y) \log(1 - y')$$

x1

x2

Y'

$$w_1 x_1 + w_2 x_2 + b_1 = z$$

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

Error Function

$$= \quad -y \log(y') - (1-y) \log(1-y')$$
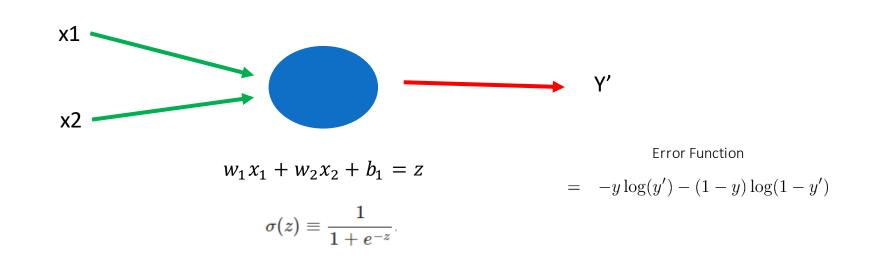
Back Propagation

$Step\ 1\ -\ $Initialization

W1	W2	B
2	3	-4

Gradient Descent

Back Propagation

x1

x2

Y'

$$w_1 x_1 + w_2 x_2 + b_1 = z$$

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

Error Function

$$= \quad -y \log(y') - (1 - y) \log(1 - y')$$

$Step\ 2\ -$ Forward propagation
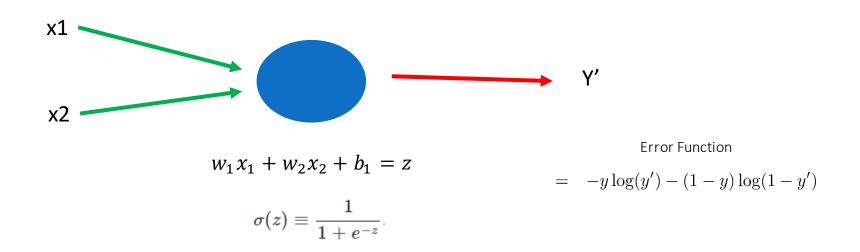
x1	x2	y
10	-4	1

$$z = 2 \times 10 + 3 \times -4 + (-4) = 4$$

Applying activation function $\sigma(z) = 0.982$

Gradient Descent

Back Propagation

x1

x2

Y'

$$w_1 x_1 + w_2 x_2 + b_1 = z$$

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

Error Function

$$= -y \log(y') - (1-y) \log(1-y')$$
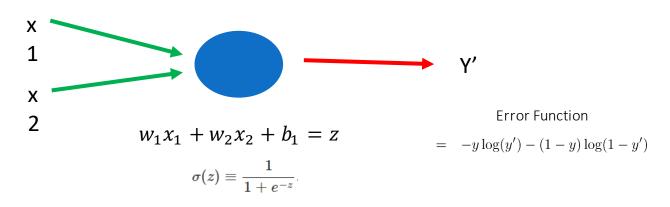
$$Step\ 3 - \text{Error calculation} \quad = \quad -y \log(y') - (1-y) \log(1-y')$$

Y'	y
0.982	1

$$E = 0.0079$$

Gradient Descent

x
1

x
2

$$w_1 x_1 + w_2 x_2 + b_1 = z$$

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

Y'

Error Function

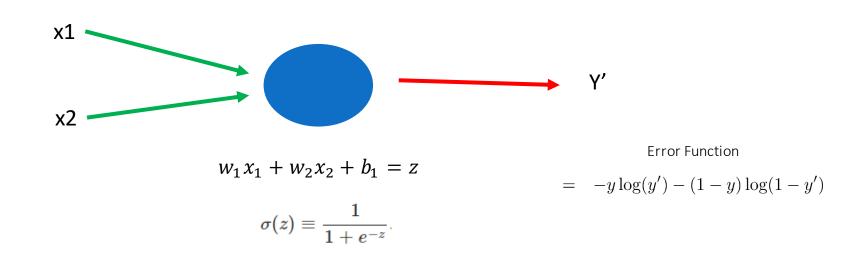$$= -y \log(y') - (1 - y) \log(1 - y')$$

Back Propagation

$Step\ 4 -$ Back Propagation

$$\frac{\partial E}{\partial y'} = \text{slope of error wrt y}' = \frac{\partial(-1 \times \log(y')\ )}{\partial y'} = -\frac{1}{y'}$$

$$\frac{\partial y'}{\partial z} = \text{slope of activation function wrt z} = \frac{e^{-z}}{(1 + e^{-z})^2}$$

$$\frac{\partial z}{\partial w_1} = x_1 = 10 \qquad \frac{\partial z}{\partial w_2} = x_2 = -4 \qquad \frac{\partial z}{\partial b} = 1$$

Gradient Descent

Back Propagation

$Step\ 4\ -$ Back Propagation

$$\frac{\partial E}{\partial y'} = \text{slope of error wrt y}' = \frac{\partial(-1 \times \log(y'))}{\partial y'} = -\frac{1}{y'}$$

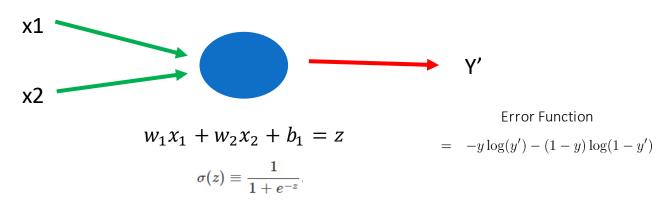$$\frac{\partial y'}{\partial z} = \text{slope of activation function wrt z} = \frac{e^{-z}}{(1 + e^{-z})^2}$$

$$\frac{\partial z}{\partial w_1} = x_1 = 10 \qquad \frac{\partial z}{\partial w_2} = x_2 = -4 \qquad \frac{\partial z}{\partial b} = 1$$

$To\ get\ \frac{\partial E}{\partial w_1}\ i.e.\ \Delta w_1$ we apply chain rule $\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial y\prime} \times \frac{\partial y\prime}{\partial z} \times \frac{\partial z}{\partial w_1} = -0.186$

$Similarly\ \frac{\partial E}{\partial w_2} = 0.0746 \qquad \frac{\partial E}{\partial b} = -0.0186$

Gradient Descent

$$w_1 x_1 + w_2 x_2 + b_1 = z$$

Error Function

$$= -y \log(y') - (1 - y) \log(1 - y')$$

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}.$$
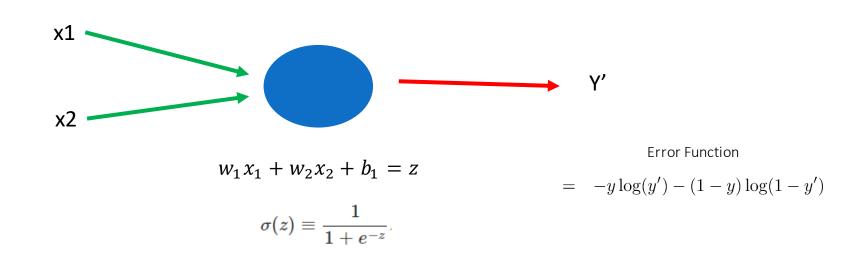
Back Propagation

$Step\ 5 -$ Updating w and b

$w1 = w1 - \alpha \Delta w1 = 2 - 5 \times -0.186 = 2.93$

$w2 = w2 - \alpha \Delta w2 = 3 - 5 \times 0.0746 = 2.627$

$b = b - \alpha \Delta b = -4 - 5 \times -0.0186 = -3.907$

Gradient Descent

x1

x2

Y'

$$w_1 x_1 + w_2 x_2 + b_1 = z$$

$$= \quad -y \log(y') - (1-y) \log(1-y')$$

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

Back Propagation

$Repeat\ Step\ 2\ -$

x1	x2	y
10	-4	1

$$z = 2.9 \times 10 + 2.6 \times -4 + (-3.9) = 14.7$$

Applying activation function $\sigma(z) = 0.999$