Human Brain VS Computer

Motivation

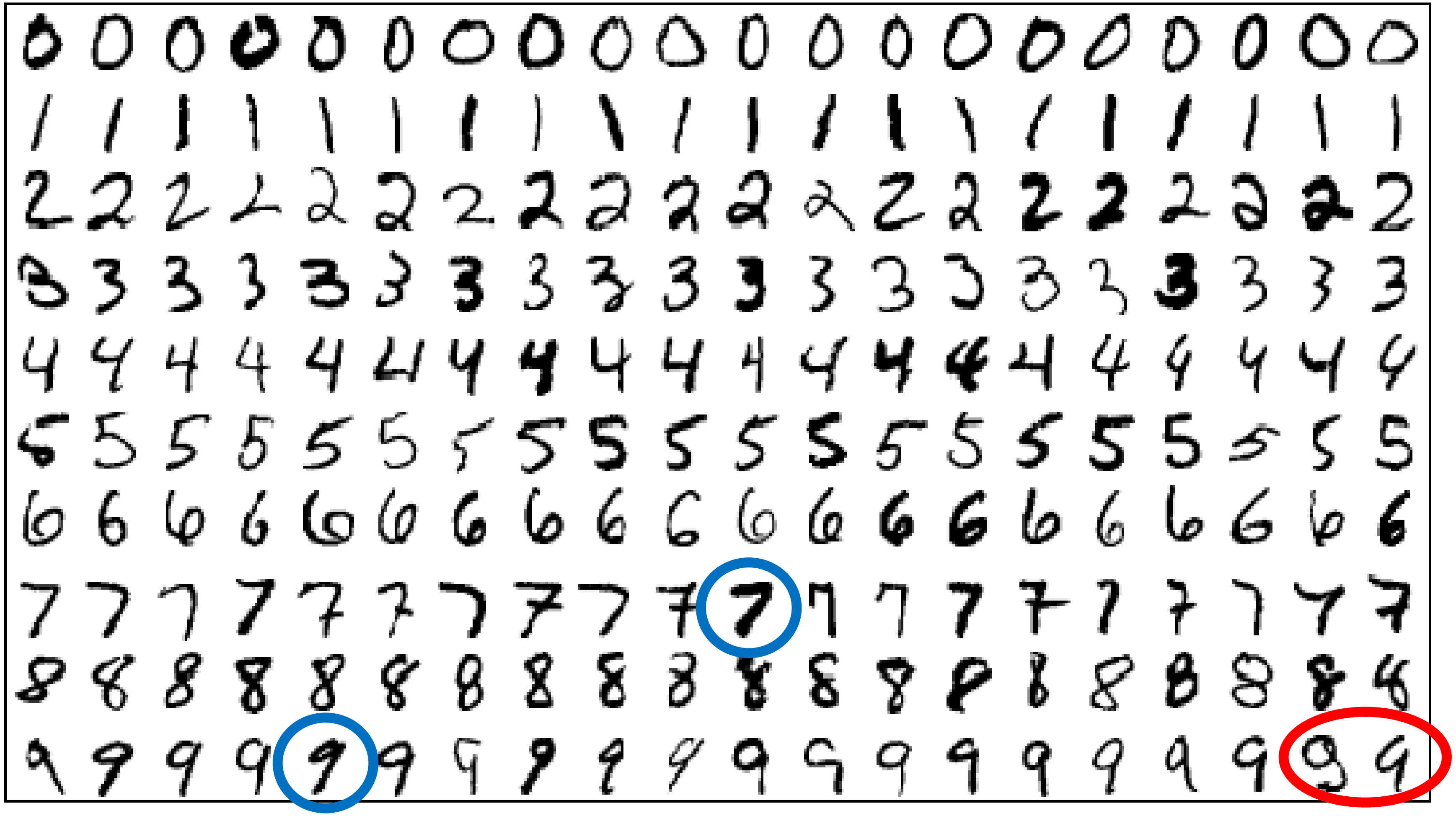
Human mind - Good at image recognition, pattern recognition etc

Computer - Good at arithmetic calculations

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

Handwriting recognition

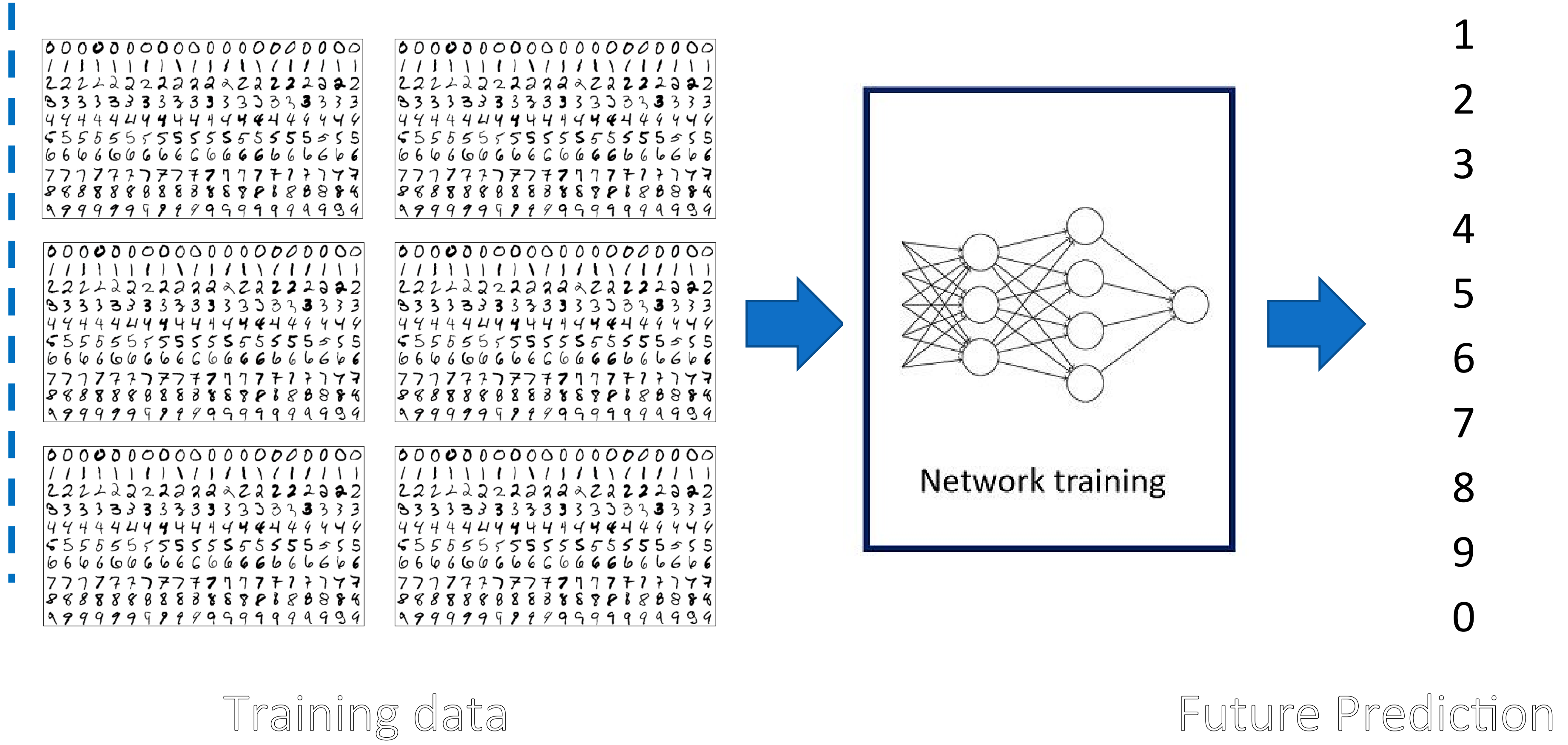
Making precise rules is difficult



Neural Networks

Neural Networks creates own complex pattern recognition rules

Pattern recognition



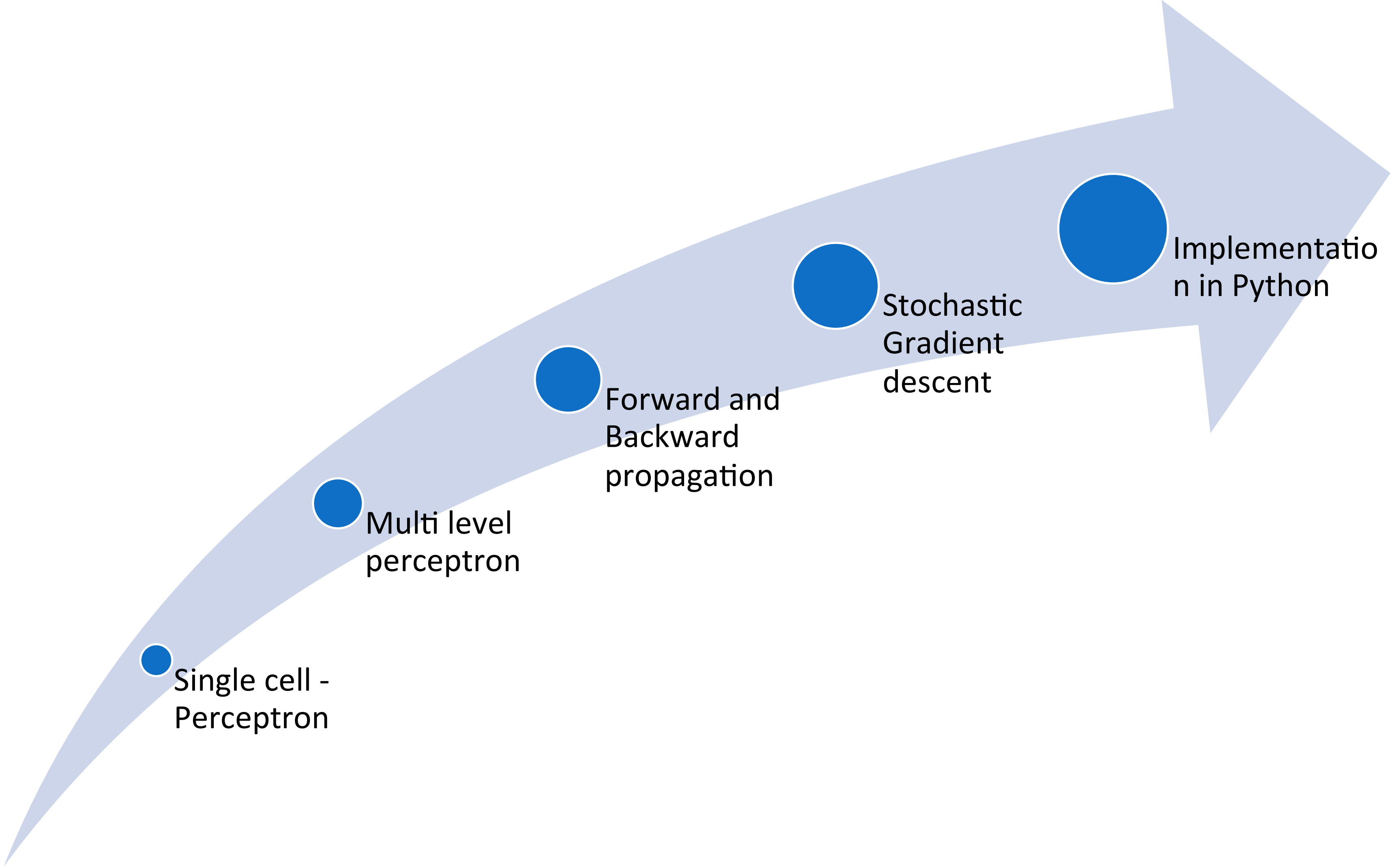
Dataset

Fashion MNIST

We will classify images into 10 fashion items

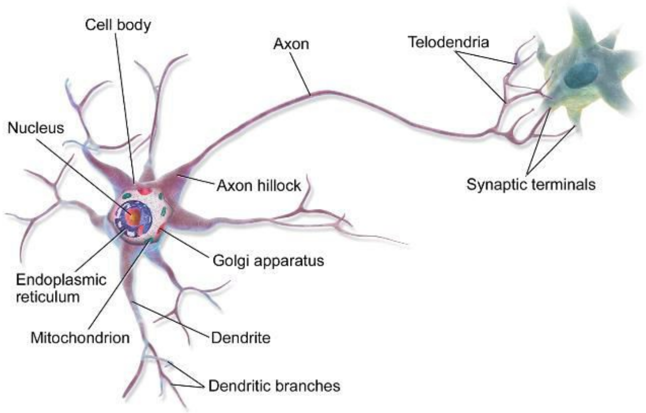
Course Flow

Course Flow

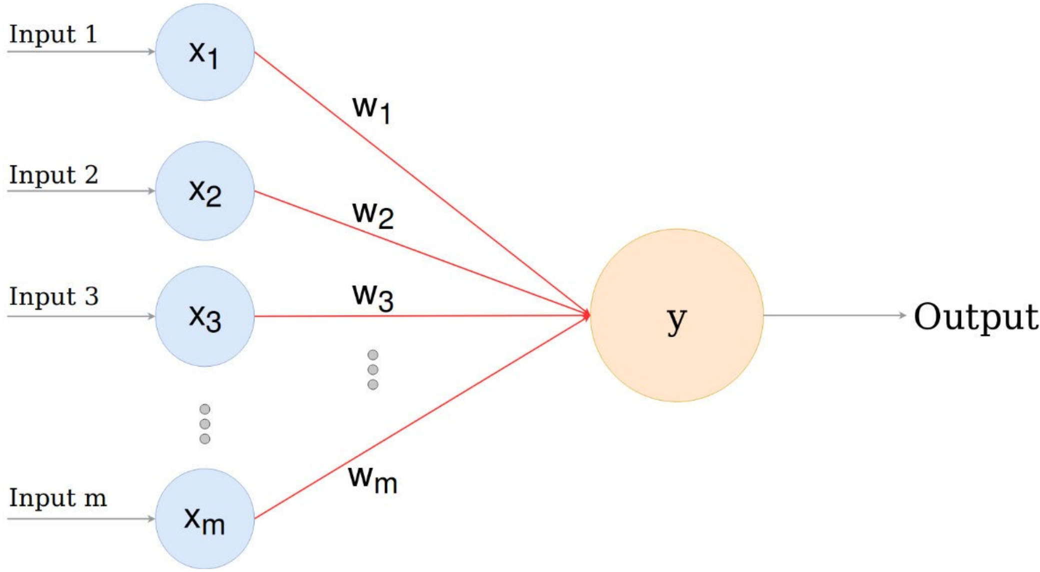


Perceptron

Artificial Neuron



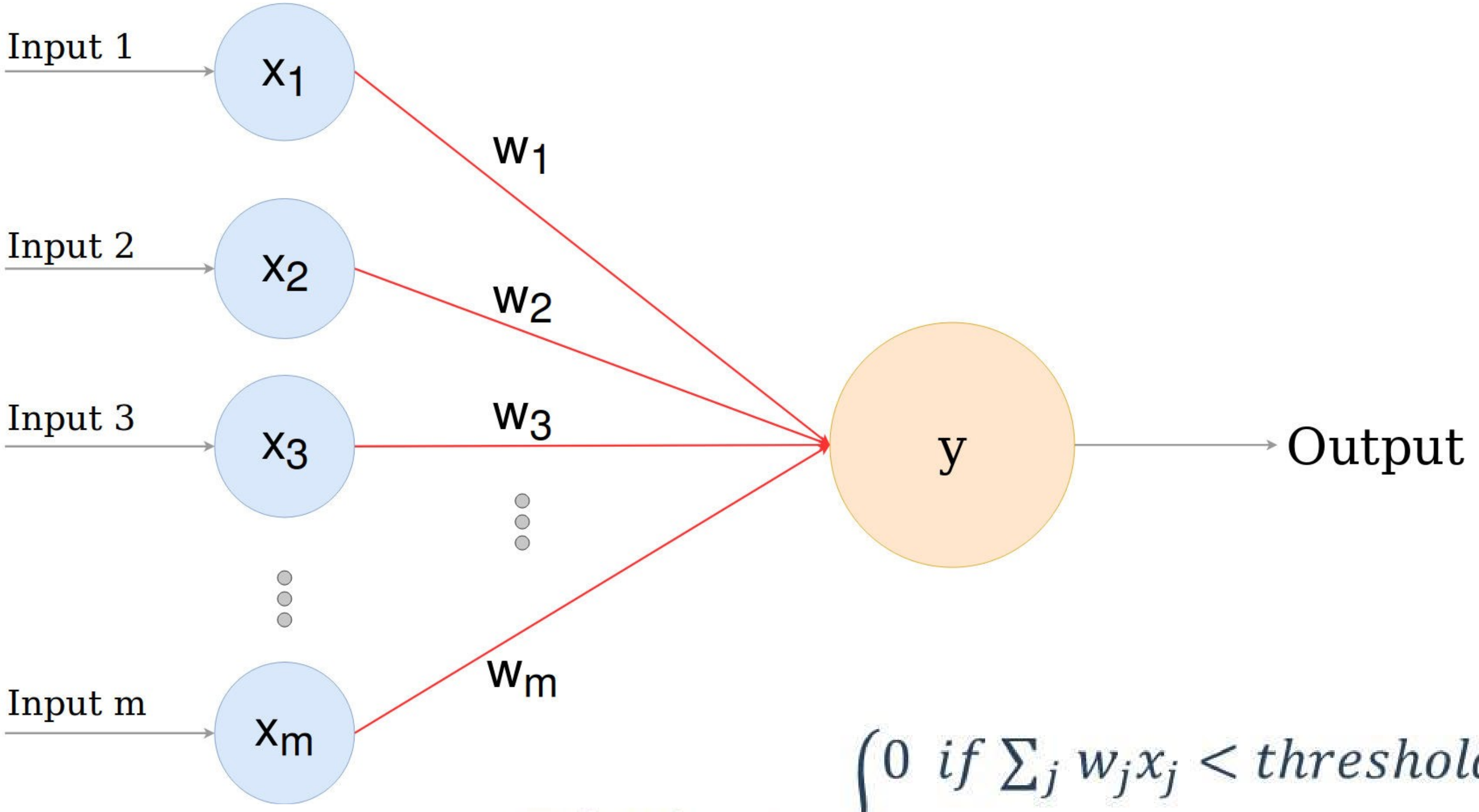
Biological Neuron



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

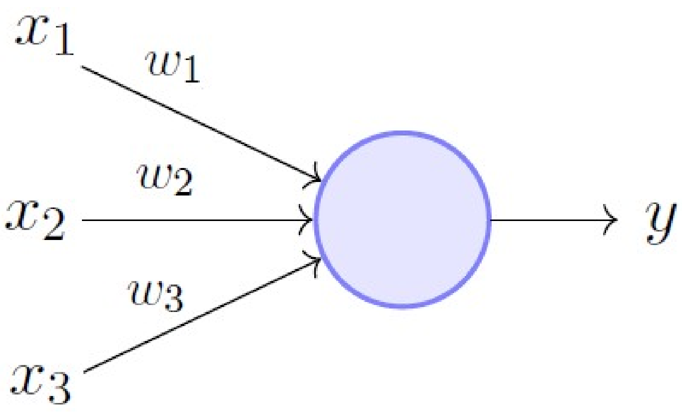
- Blue or Not

Sleeves

- Full or half

Fabric

- Cotton or not



You will buy the shirt or not

Example

Purchasing a Shirt

Color

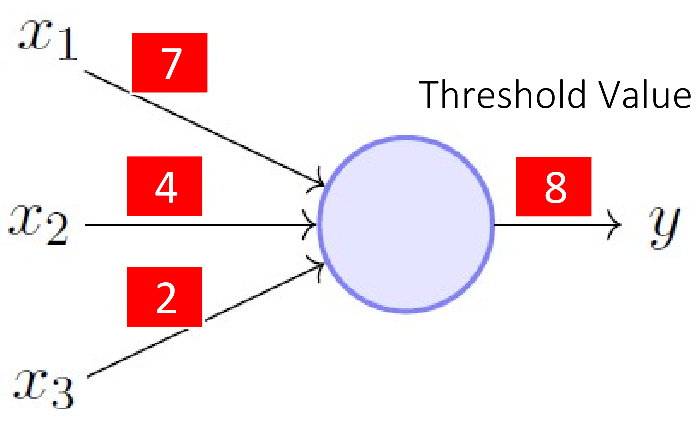
- Blue or Not

Sleeves

- Full or half

Fabric

- Cotton or not



You will buy the shirt or not

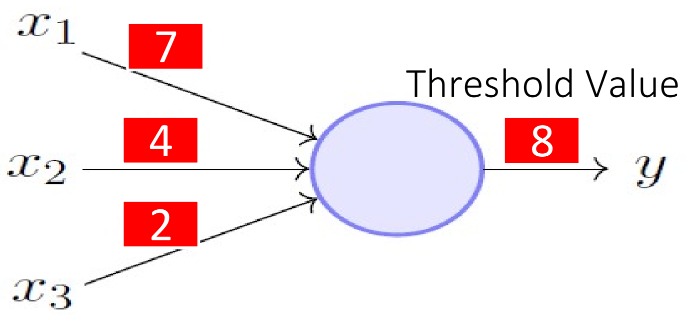
Example

Purchasing a Shirt

Color
- Blue or Not

Sleeves
- Full or half

Fabric
- Cotton or not



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

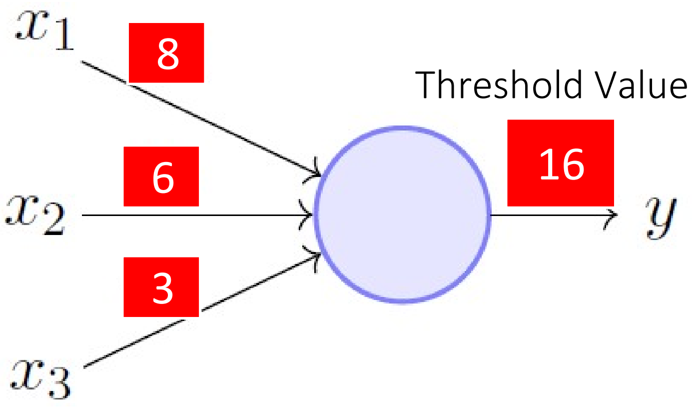
Example

Purchasing a Shirt

Color
- Blue or Not

Sleeves
- Full or half

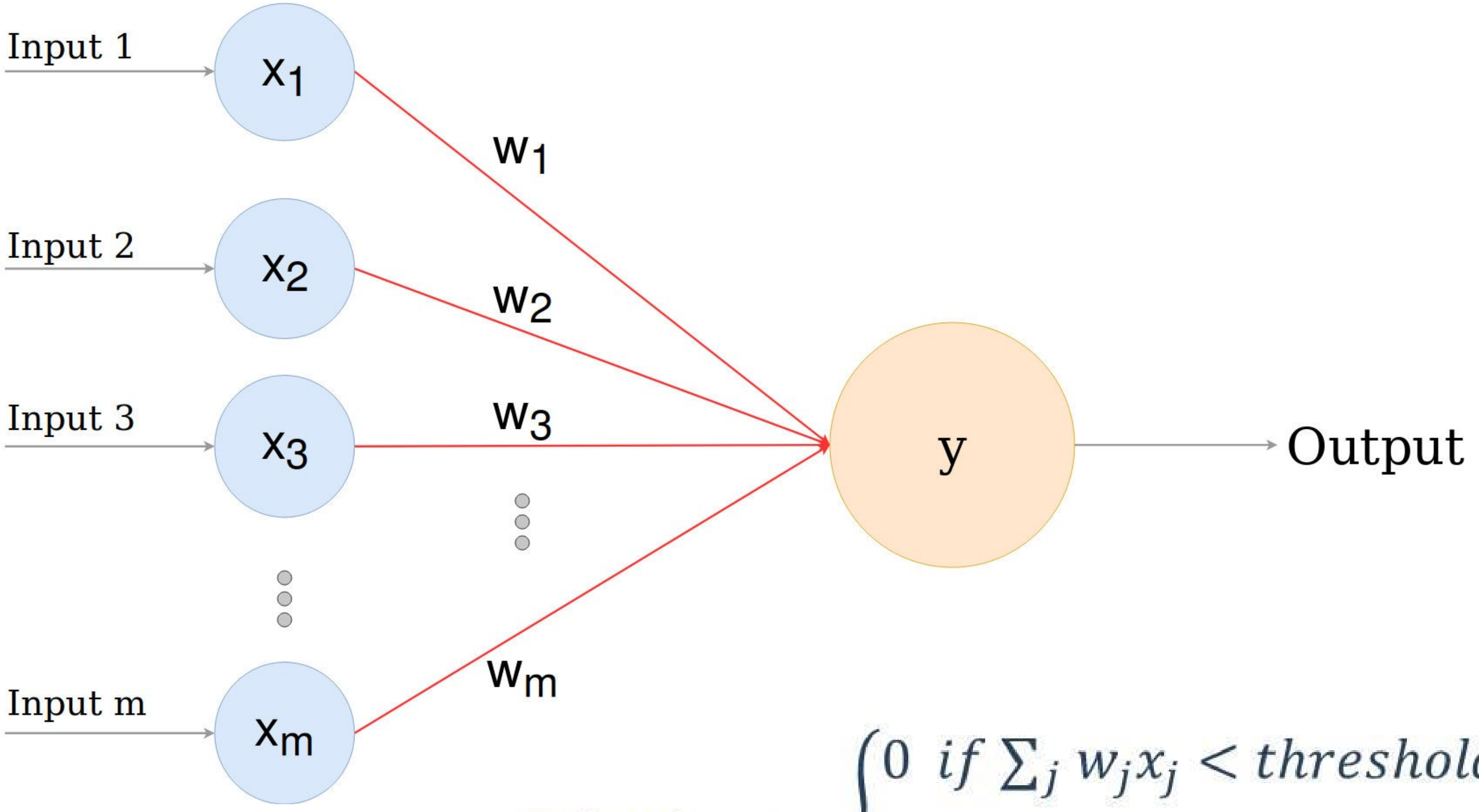
Fabric
- Cotton or not



You will buy the shirt or not

Perceptron

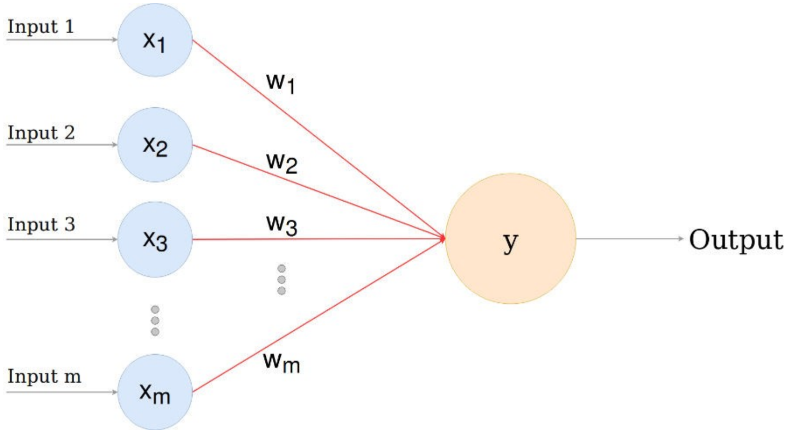
Removing Binary Restriction



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

Perceptron

Standard Equation



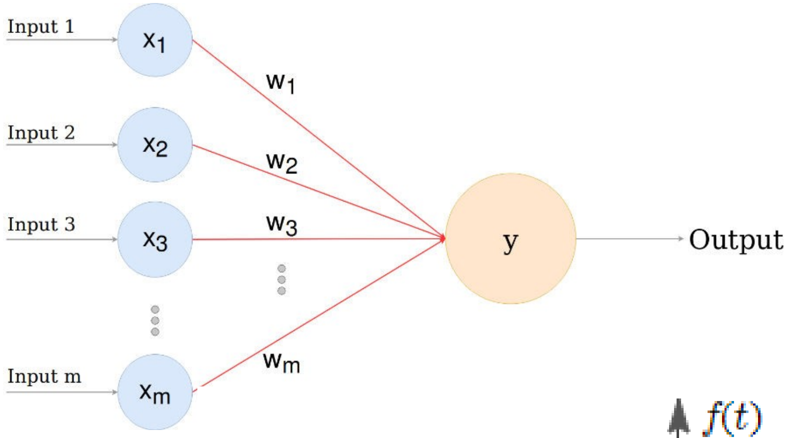
$$output = \begin{cases} 0 \ if \sum_j w_j x_j < threshold \\ 1 \ if \sum_j w_j x_j > threshold \end{cases}$$

$$Output = \begin{cases} 0, & w_{j} \ {}_{j} \, x_j + b < 0 \\ 1, & w_{j} \ {}_{j} \, x_j + b \geq 0 \end{cases}$$

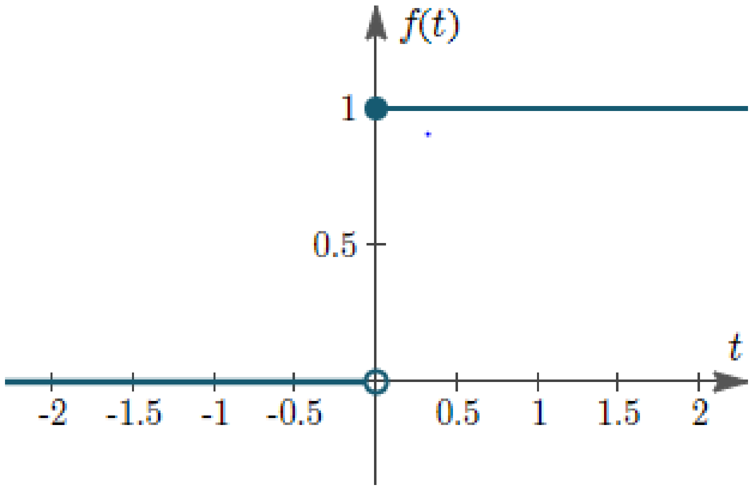
b is called Bias

Perceptron

Graphical Representation



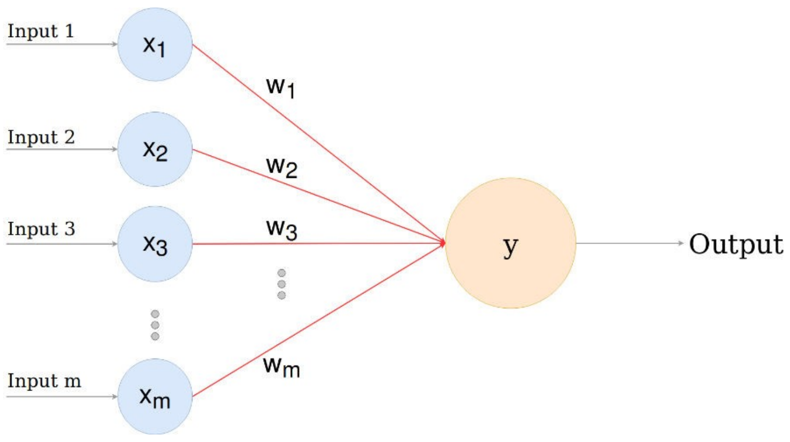
$$Output = \begin{cases} 0, & \sum_j w_j x_j + b < 0 \\ 1, & \sum_j w_j x_j + b \geq 0 \end{cases}$$



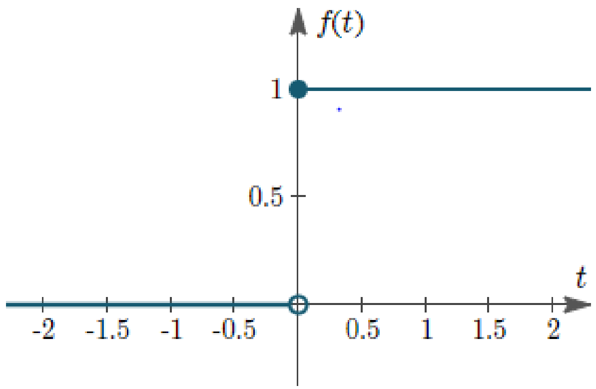
Step Activation function

Perceptron

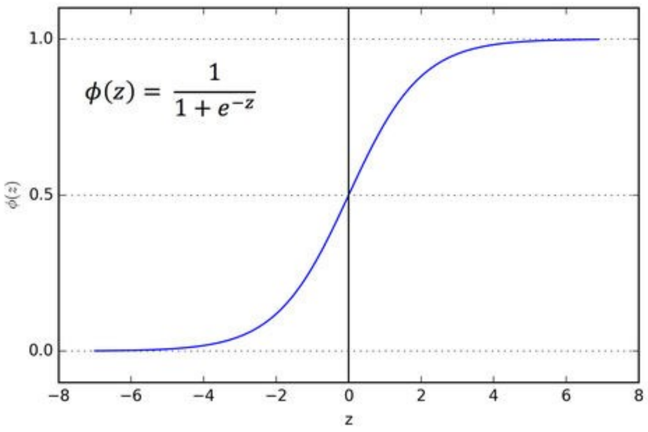
Sigmoid Activation



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



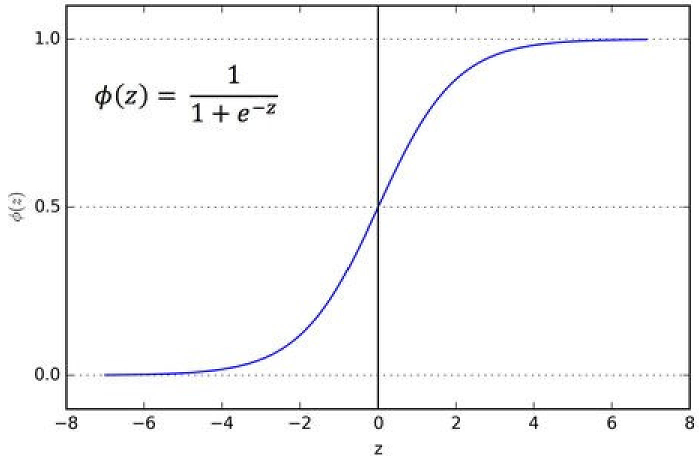
Step Activation function



Sigmoid Activation function

Perceptron

Sigmoid Activation



Sigmoid Activation function

- Sigmoid is better because it is less sensitive to individual observation
- Artificial neuron with sigmoid activation is called sigmoid or logistic neuron

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

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

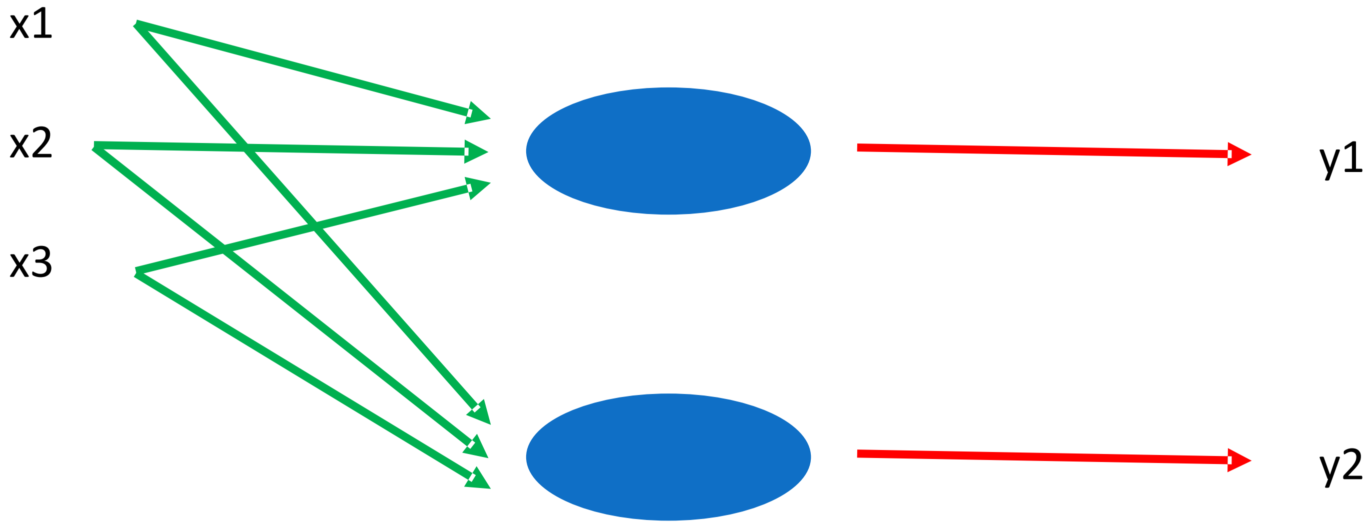
Making Networks

Two types of Stacking

- Parallel
- Sequential

Making Networks

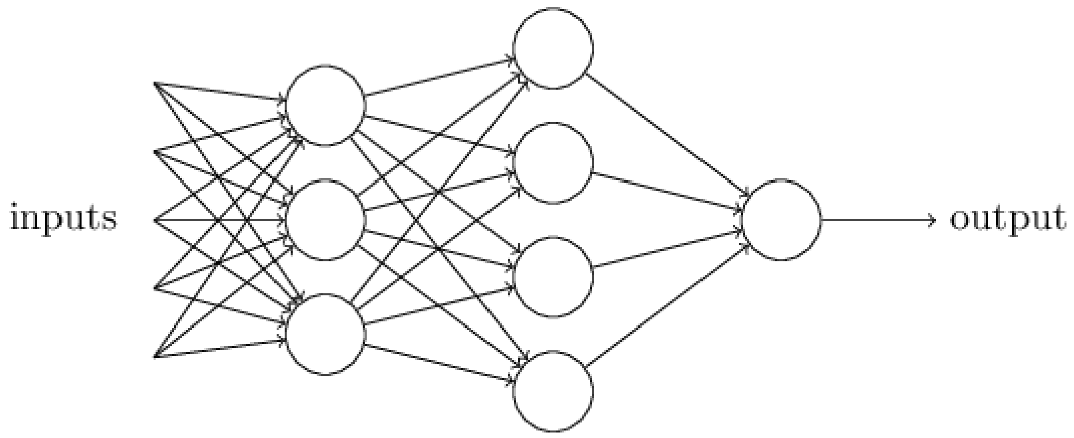
Parallel Stacking



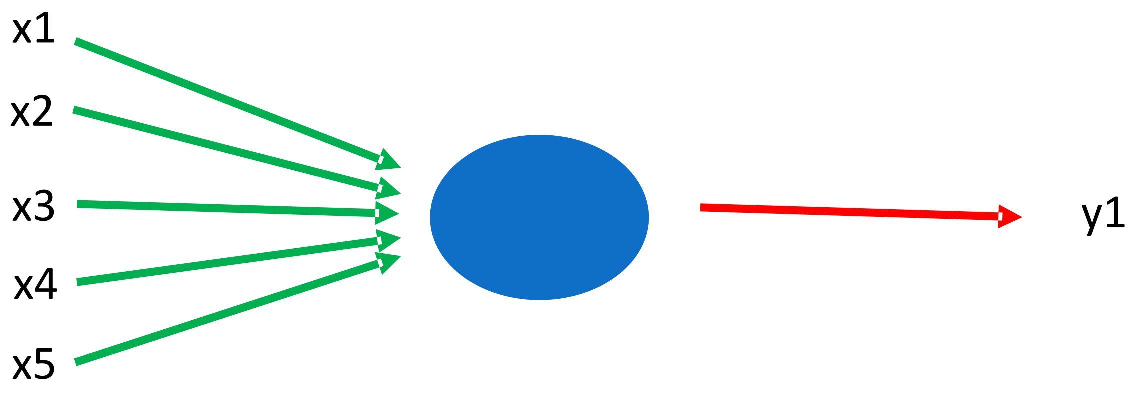
With parallel stacking we can get multiple outputs with the same input

Making Networks

Sequential Stacking

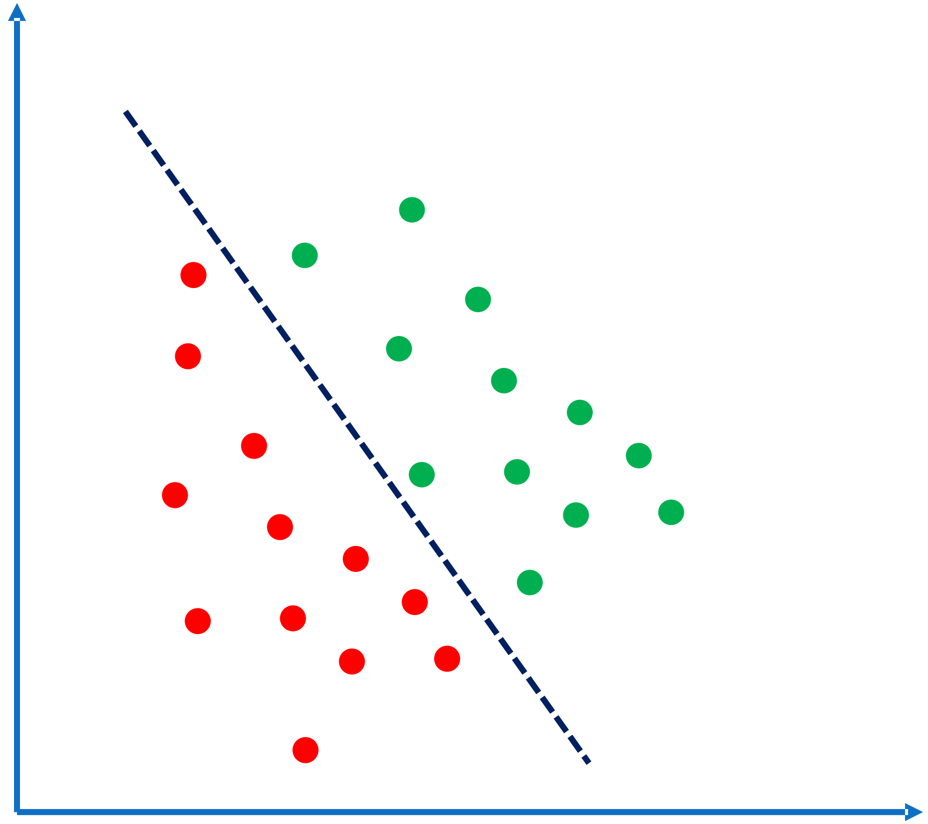


Why not use a single neuron



Making Networks

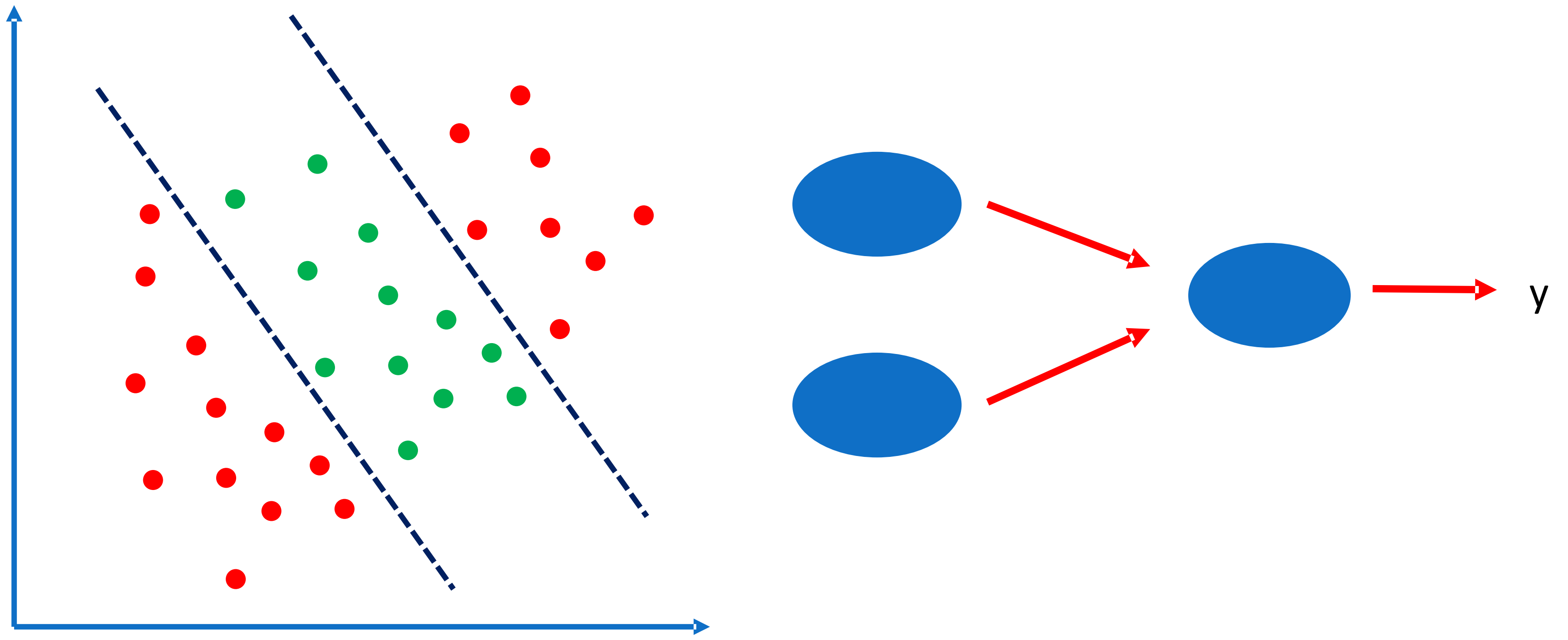
Sequential Stacking



Single neuron can handle such linear classification problem

Making Networks

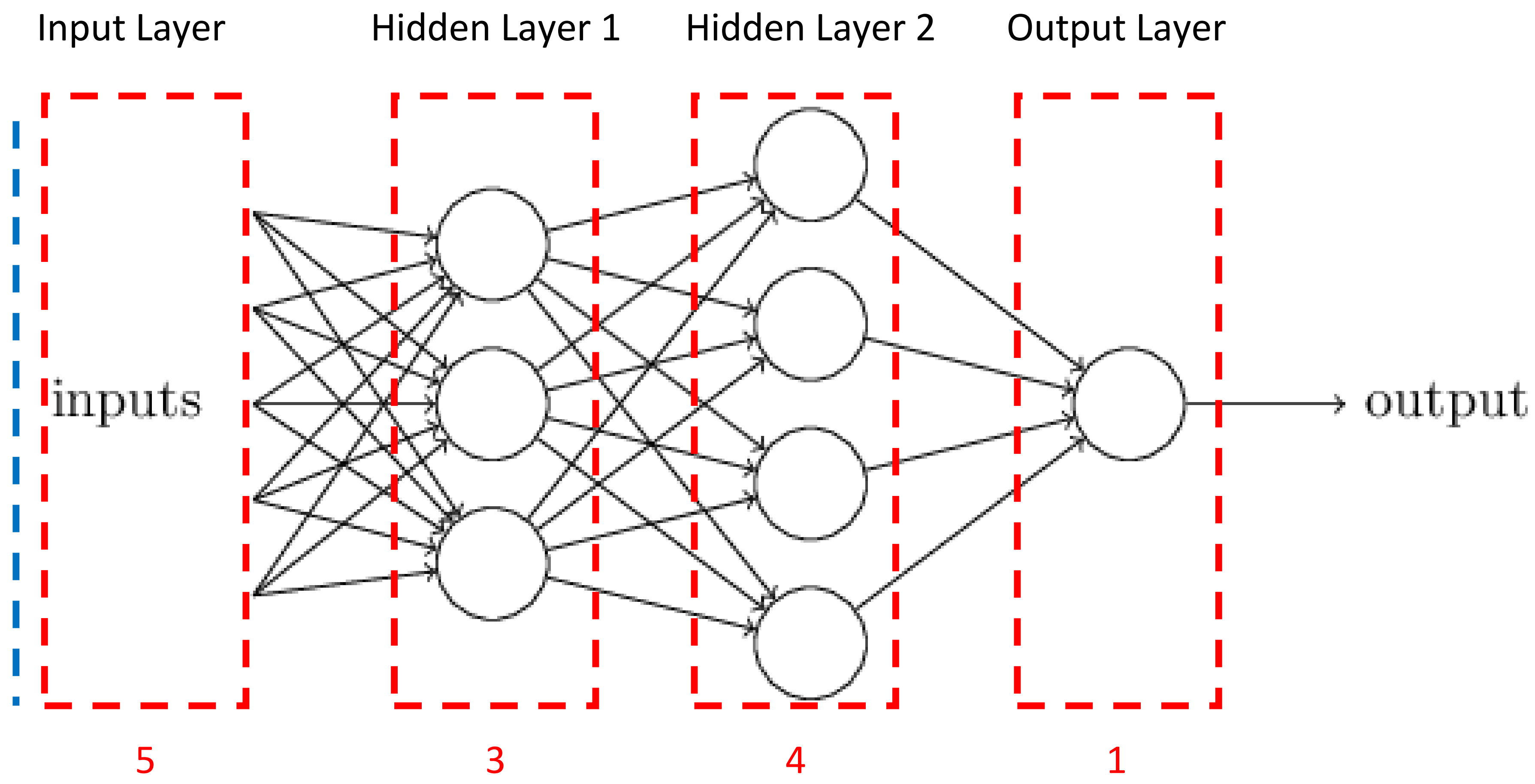
Sequential Stacking



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

Making Networks

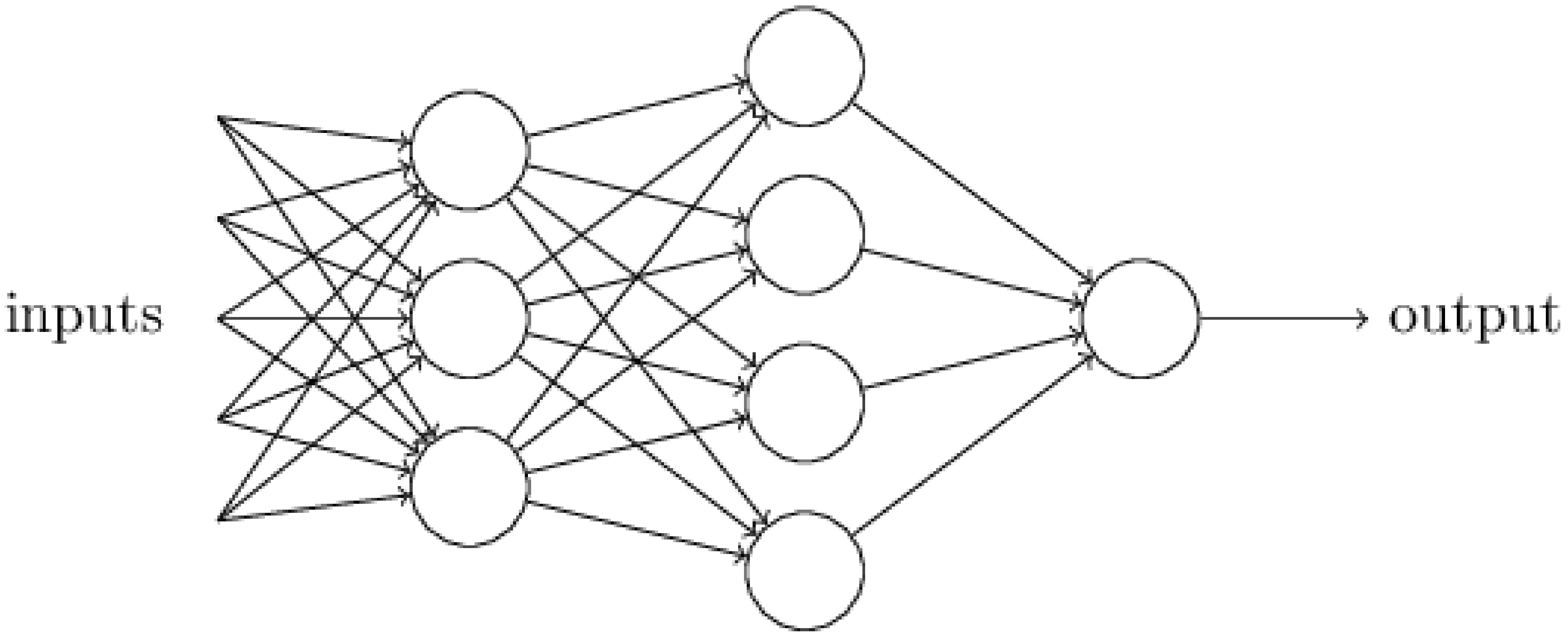
Nomenclature



5-3-4-1 Network

Making Networks

Nomenclature



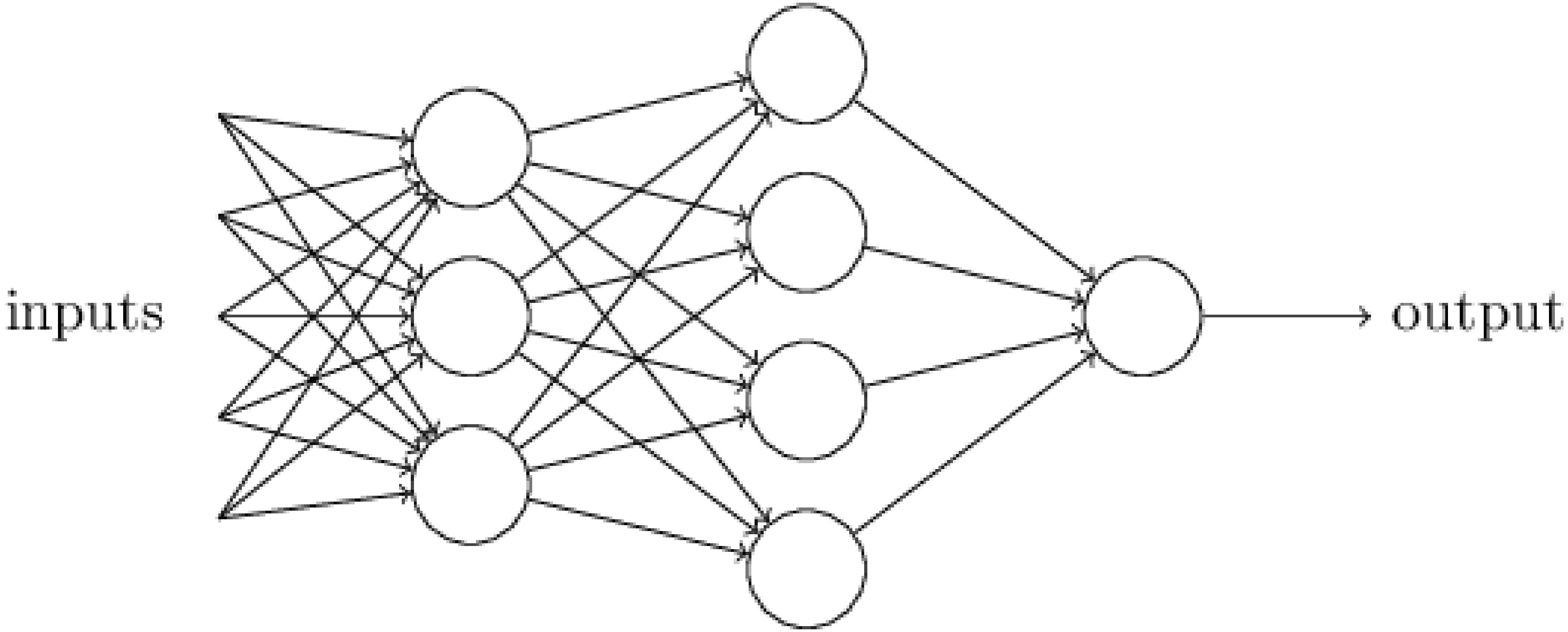
Feed Forward Network – One directional processing

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



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

Neural Network

How it works

Covered till Now

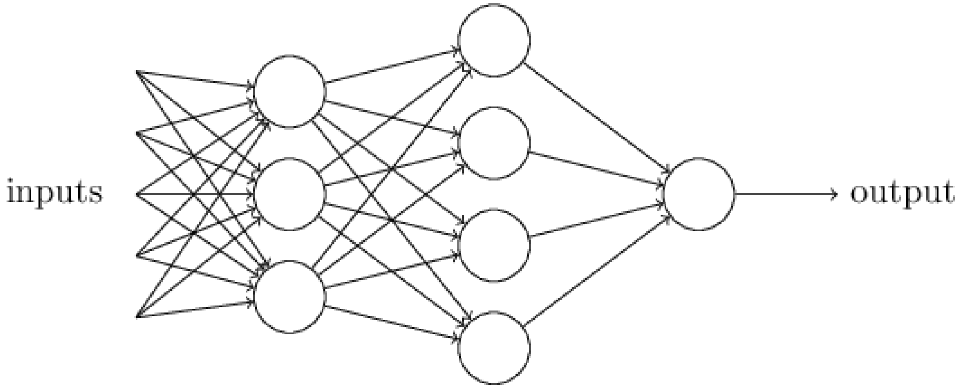
- What is a neural network

Now we are going to learn

- How does a neural network works

Problem Statement

Quick Recap



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

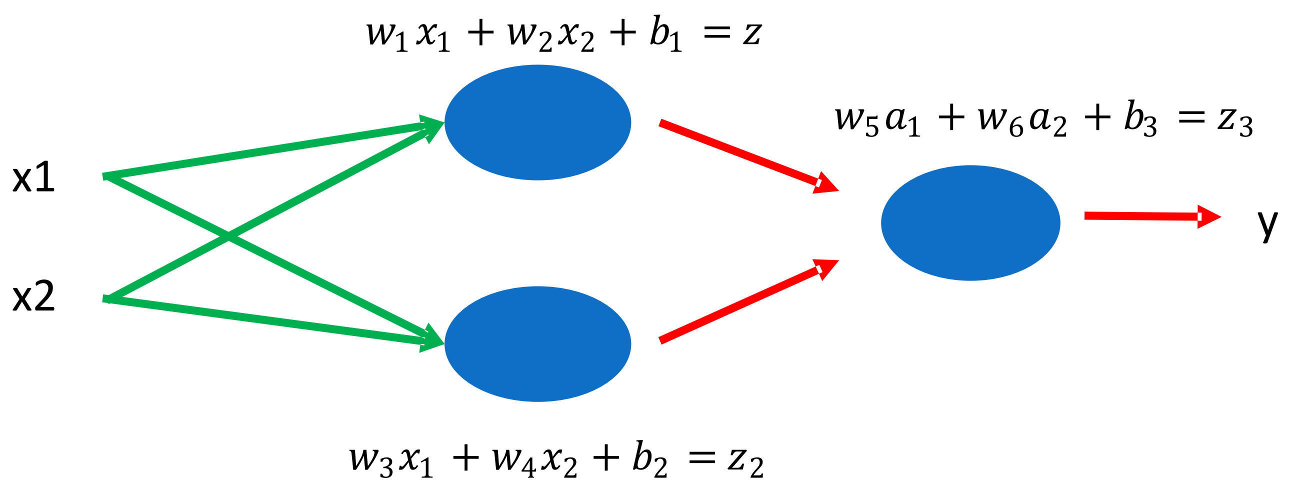
$$Output = \frac{1}{1+\exp(-\sum_j w_j x_j - b)}.$$

Problem Statement

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

Problem Statement

Example



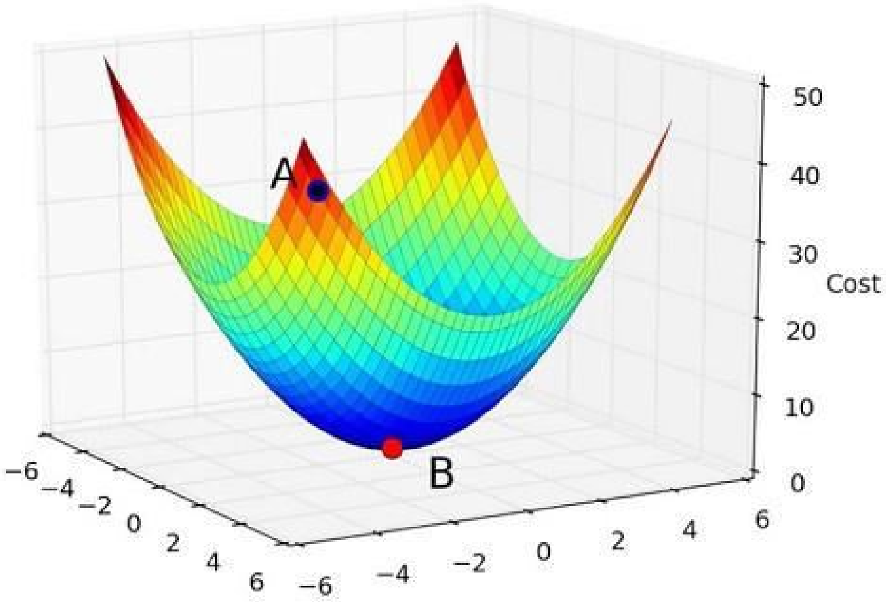
Variables to be established in this neural network

- Weights - W1, W2..........W6
- Biases - B1, B2, B3

Total - 9 variables

Neural Network

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

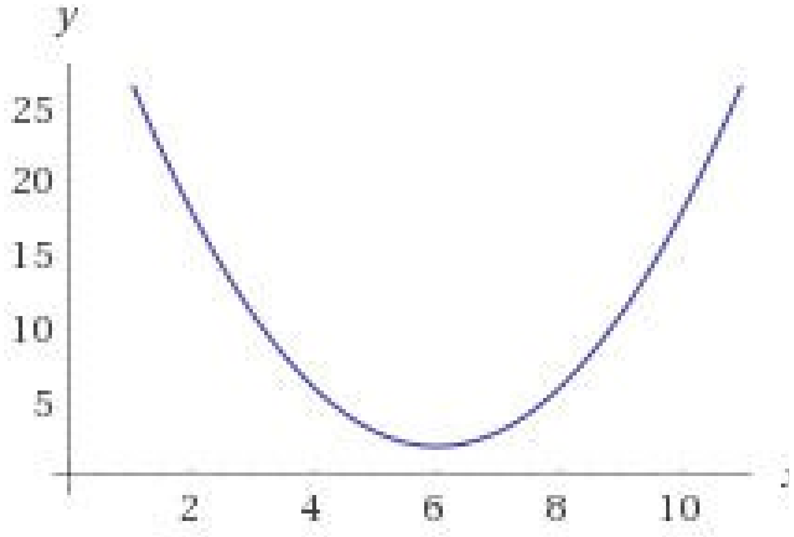
Gradient Descent

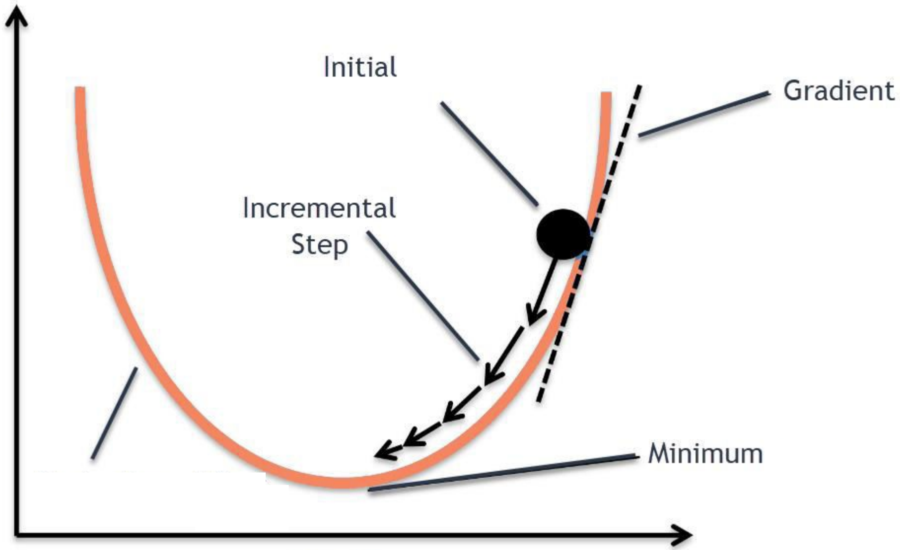
Process

- **Step 1**: Assign random W and B values — Initialization
- **Step 2**: Calculate final output using these values — Forward Propagation
- **Step 3**: Estimate error using error function — Backward Propagation
- **Step 4**: Find those W and B which can reduce this error — Backward Propagation
- **Step 5**: Update W and B and repeat from step 2 — Implementation of GD

Neural Network

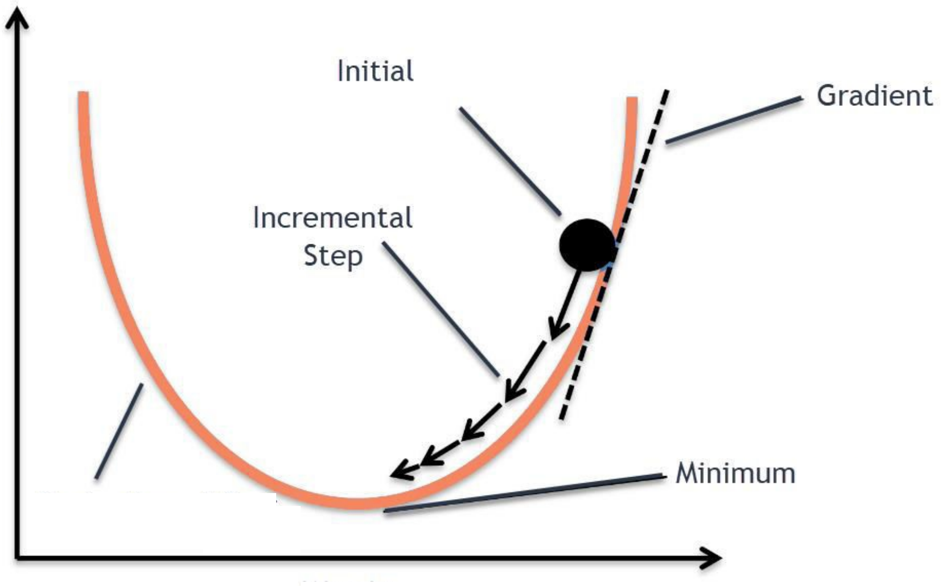
Gradient Descent





Neural Network

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

Error Function

- 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

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

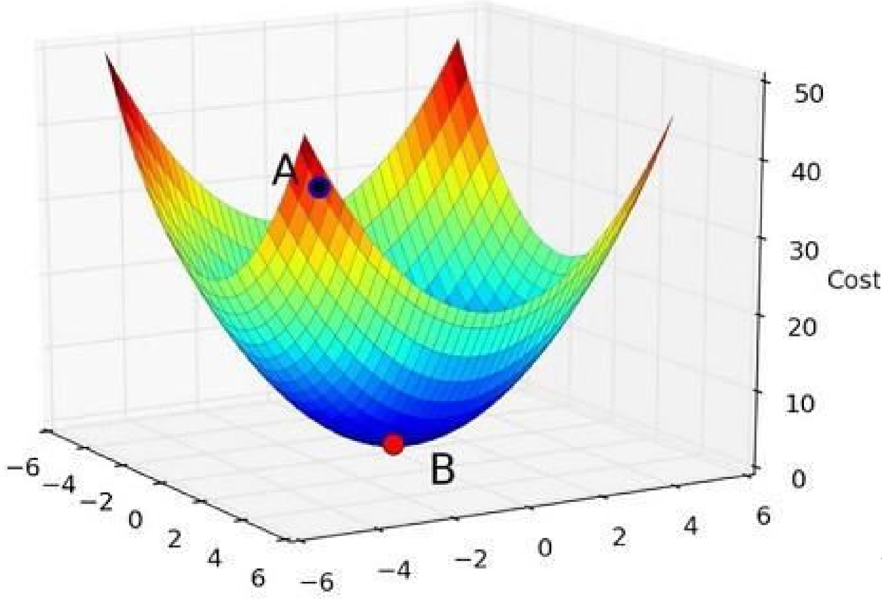
Square function works well with regression but not with classification

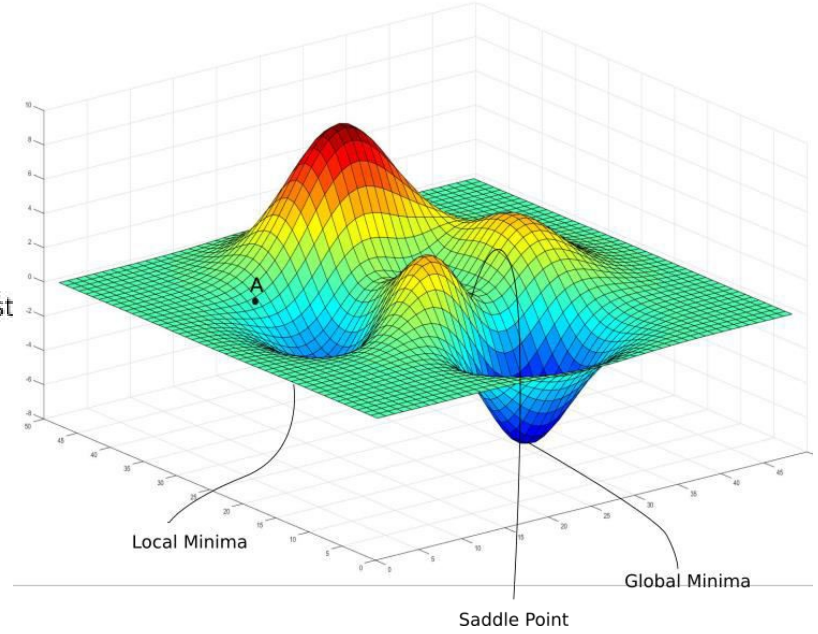
Gradient Descent

Error Function

Cross Entropy Error Function

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





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’)

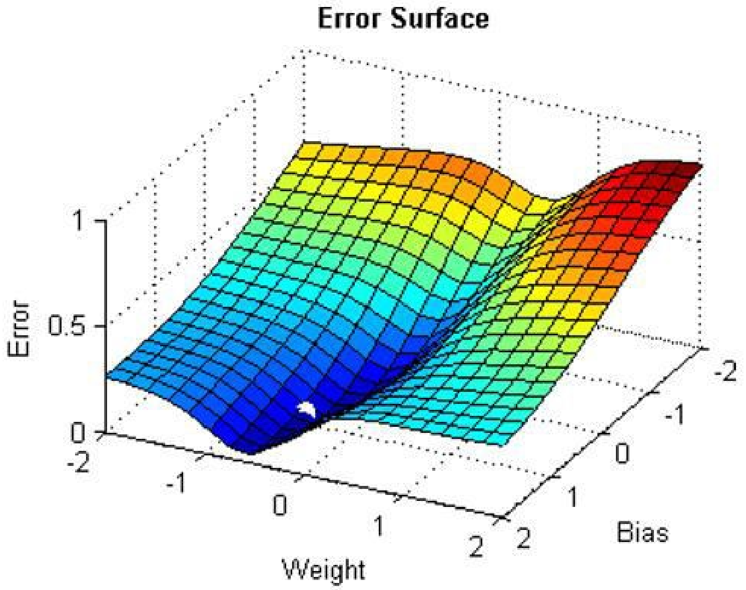
$\Rightarrow$ Maximize y’

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



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

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

α is learning rate, Δw *and* Δb are unit steps

Alpha determines number of steps we take in downward direction

Gradient Descent

Back Propagation

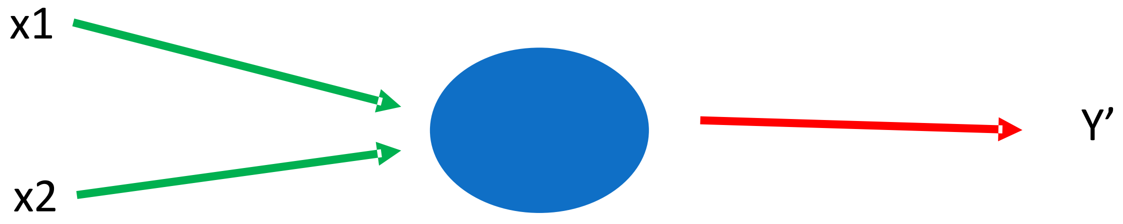
$w = w - \alpha \Delta w$

$b = b - \alpha \Delta b$

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

We do back propagation

Example



$w_1 x_1 + w_2 x_2 + b_1 = z$

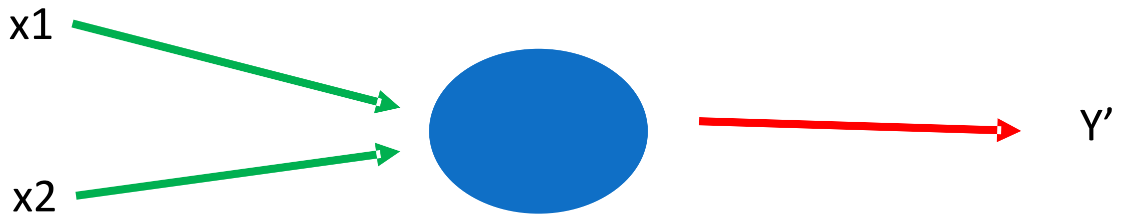
$\sigma(z) \equiv \frac{1}{1 + e^{-z}}.$

Error Function

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

Gradient Descent

Back Propagation



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

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

Error Function

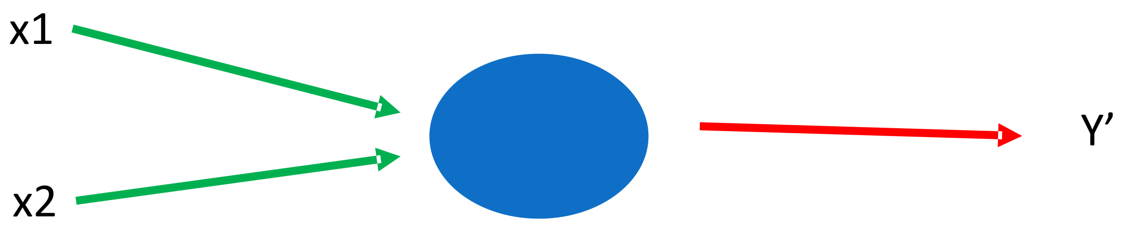
$$= -y \log(y') - (1 - y) \log(1 - y')$$

$Step\ 1$ – Initialization

W1	W2	B
2	3	-4

Gradient Descent

Back Propagation



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

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

Error Function

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

$Step\ 2$ – Forward propagation

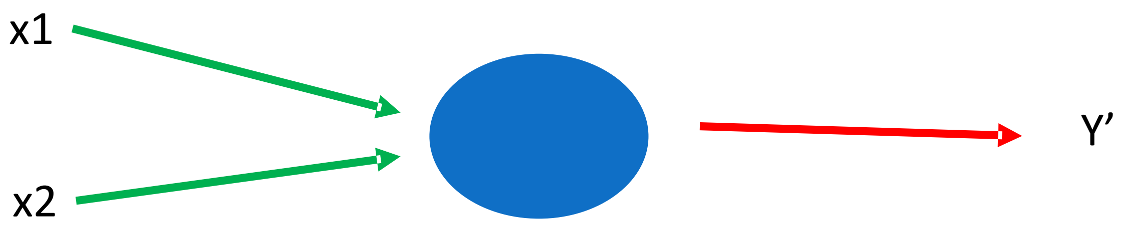
x1	x2	y
10	-4	1

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

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

Gradient Descent

Back Propagation



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

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

Error Function

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

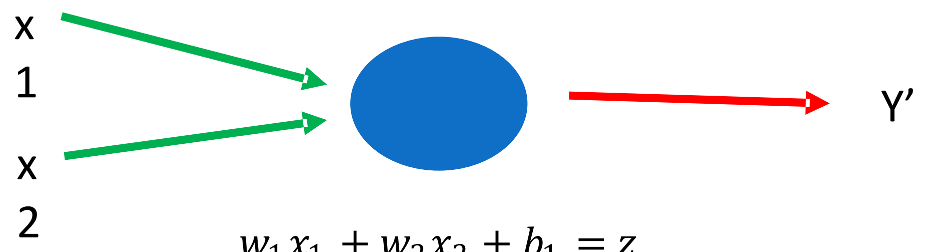
$Step\ 3$ – Error calculation $= -y\log(y') - (1-y)\log(1-y')$

Y’	y
0.982	1

$$E = 0.0079$$

Gradient Descent

Back Propagation



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

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

Error Function

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

$Step\ 4$ – 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

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

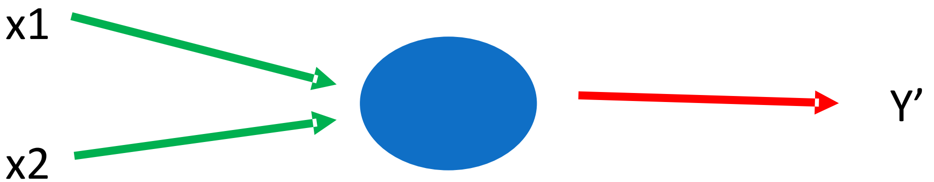
$$\frac{\partial y'}{\partial z} = \text{slope of activation function wrt z} = \frac{e^{-z}}{(1+e^{-z})^2}$$

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

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

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

Gradient Descent



$$w_1x_1 + w_2x_2 + b_1 = z$$

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

Error Function

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

Back Propagation

$Step$ 5 – Updating w and b

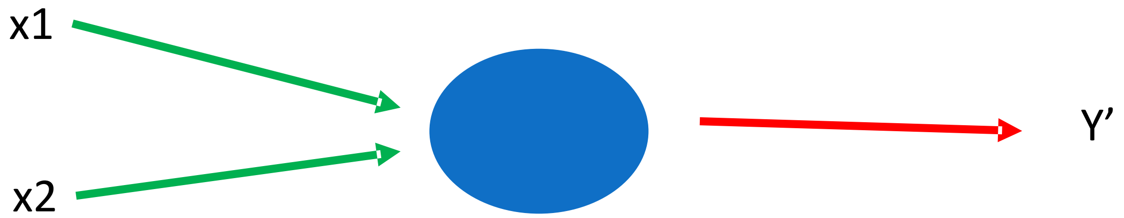
$$w1 = w1 - \alpha\Delta w1 = 2 - 5 \times -0.186 = 2.93$$

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

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

Gradient Descent

Back Propagation



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

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

Error Function

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

$Repeat\ Step\ 2\ -$

x1	x2	y
10	-4	1

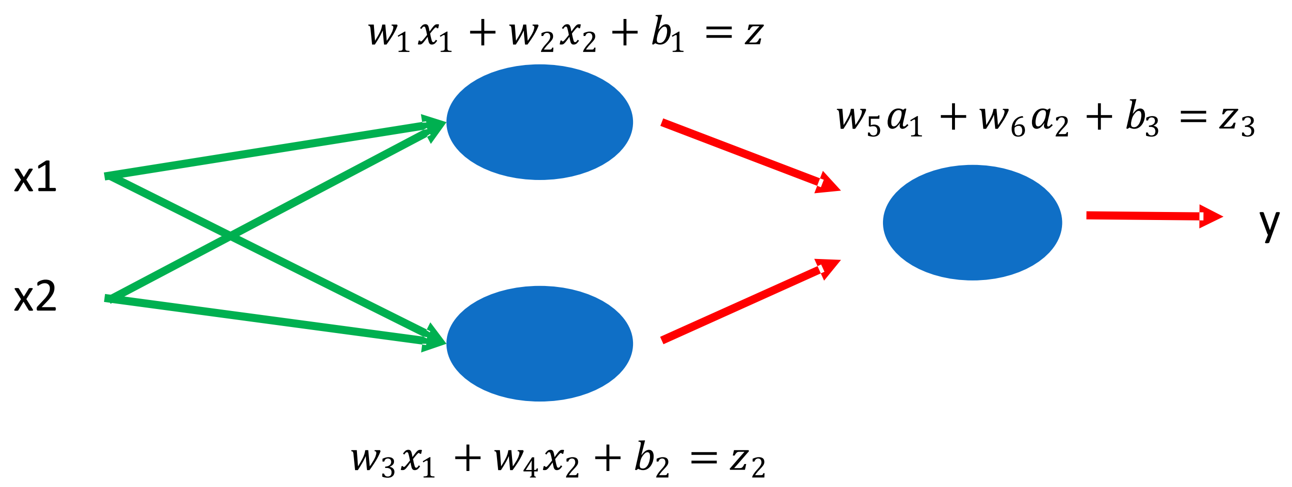
$$z = 2.9 \times 10 + 2.6 \times -4 + (-3.9) = 14.7$$

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

Neural Network

Activation Function

Q – Why do we use activation functions



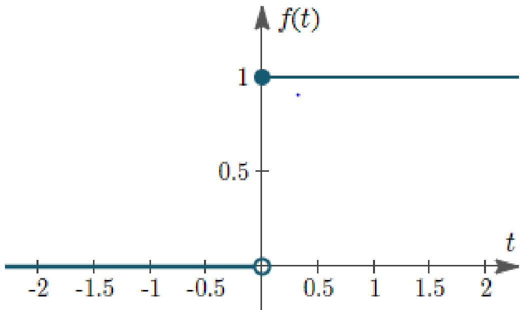
Ans

- To put special boundary conditions on the output
- To introduce non linearity and find complex patterns

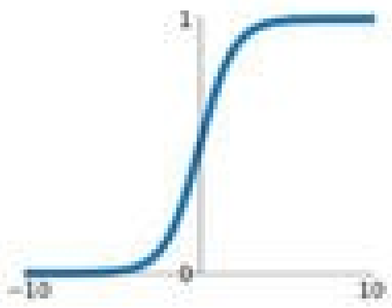
Neural Network

Activation Function

Q – What are the different types of activation functions

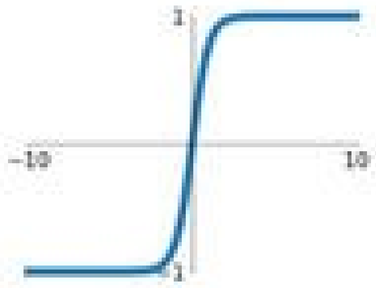


Step



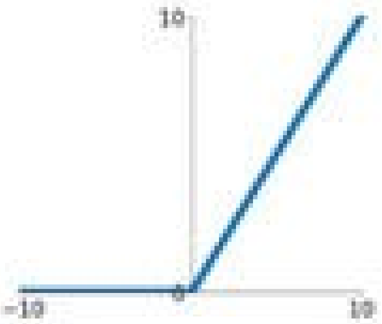
Sigmoid

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



tanh

$\tanh(x)$



ReLU

$\max(0, x)$

Neural Network

Q – What are the different types of activation functions

Activation Function

Function	Upper Boundary	Lower Boundary	Class /Reg	Layer
Step	1	0	Classification	Mostly Output
Sigmoid	1	0	Classification	Hidden & Output
Hyperbolic Tangent (TanH)	1	-1	Classification	Hidden & Output
Rectified Linear Unit (ReLU)	0	infinity	Regression/ classification	Hidden

Neural Network

Activation Function

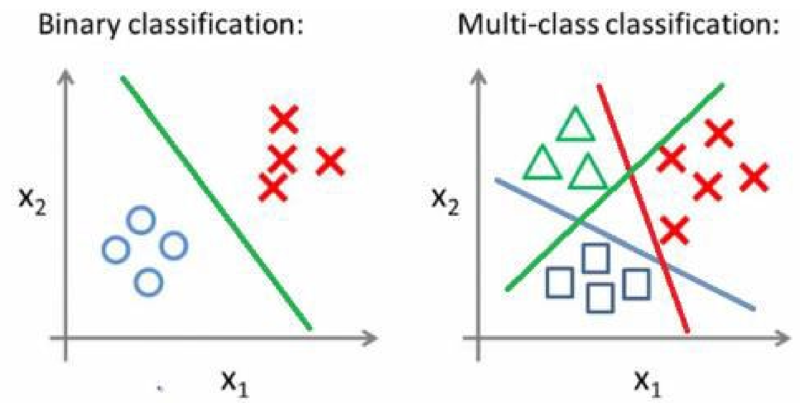
Q – Can Hidden layers and output layers have different activation functions?

Ans - Yes

Neural Network

Activation Function

Q – What is multi class classification? Is there any specific activation function for this?



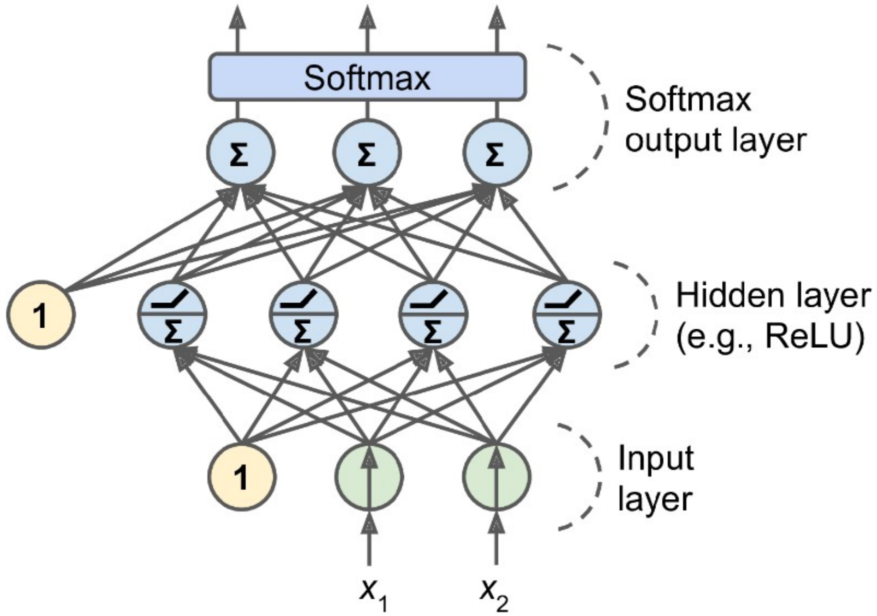
Ans

- Two classes like ‘Yes’ or ‘No’ => Binary Classification
- More than 2 classes like ‘shirts’, ‘trousers’ or ‘socks’ => Multiclass classification
- For multiclass, we use softmax activation

Neural Network

Activation Function

Q – What is multi class classification? Is there any specific activation function for this?



Ans

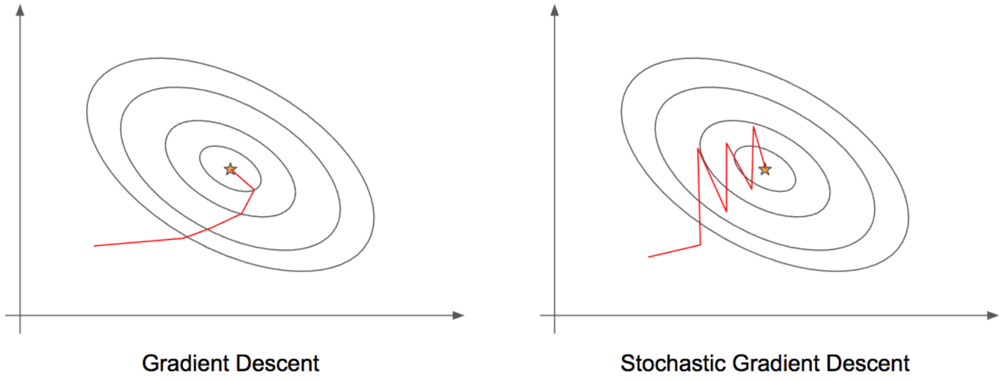
- For each class we keep one output neuron with sigmoid activation
- All the outputs go into softmax layer where each output is divided by the total sum to bring the total probability to one

Neural Network

Gradient descent

Q – What is the difference between Gradient descent and stochastic gradient descent

- Stochastic gradient descent => Single training record, forward and backward propagation
- Gradient descent => Full training set, forward and backward propagation
- Mini Batch Gradient descent => small batch of training set, forward and backward propagation



Neural Network

Epoch

Q – What is an Epoch

- Epoch is one cycle through the full training data
- It is different from iteration
- Example – Suppose we have 1000 training records, if we are doing SGD i.e. one record is input at a time, then 1000 iterations within one epoch
- If we enter 1000 records 2 time => Epoch is 2

Neural Network

Classification Hyperparameters

Hyperparameter	Typical value
# input neurons	One per input feature
# hidden layers	Depends on the problem, but typically 1 to 5
Hidden activation	ReLU

Hyperparameter	Binary classification	Multilabel binary classification	Multiclass classification
# output neurons	1	1 per label	1 per class
Output layer activation	Logistic	Logistic	Softmax
Loss function	Cross entropy	Cross entropy	Cross entropy

Neural Network

Regression Hyperparameters

Hyperparameter	Typical value
# input neurons	One per input feature
# hidden layers	Depends on the problem, but typically 1 to 5
# neurons per hidden layer	Depends on the problem, but typically 10 to 100
# output neurons	1 per prediction dimension
Hidden activation	ReLU
Output activation	None
Loss function	MSE

Neural Network

Keras & Tensorflow

Keras is a model-level library, providing high-level building blocks for developing deep-learning models

Neural Network

Keras &
Tensorflow