# Introduction to Neural Networks

## Introduction to Neural Networks

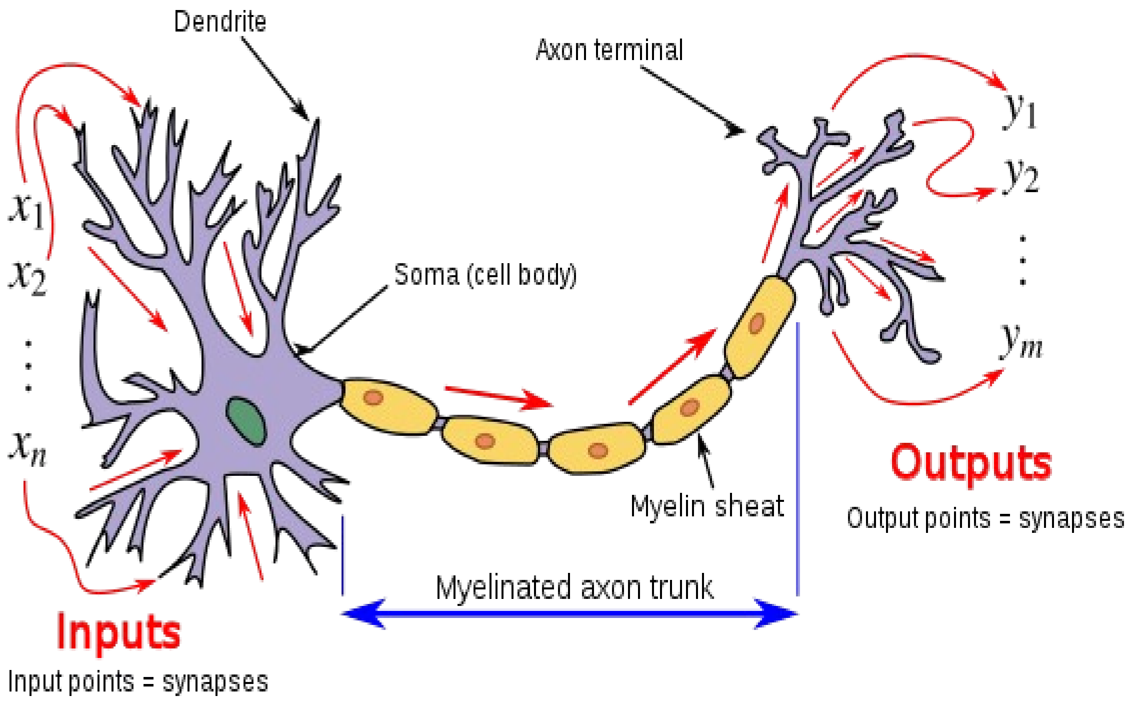
- Deep learning architectures such as recurrent neural networks(RNN) and convolutional neural networks(CNN) and deep belief networks
- have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis etc.
- The depth of computation is what has enabled deep learning models to extract the kinds of complex and hierarchical patterns found in the most challenging real world datasets

## Introduction to Neural Networks

- Deep learning models are based on artificial neural networks(ANN)
- Neural Network was inspired by the structure and function of the brain
- Biological brains are capable of solving difficult problems
- Neurons (also called as nerve cells) are the fundamental units of the human brain
- The Neurons responsible for receiving sensory input from the external world, for sending motor commands to our muscles

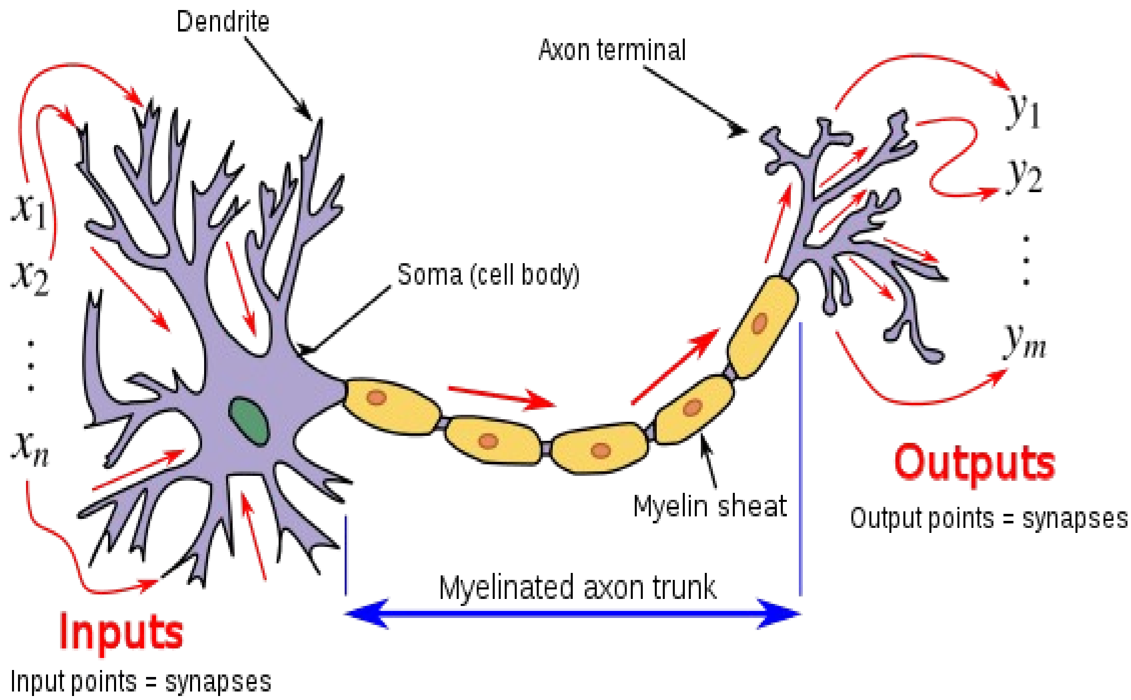
## Biological Neuron

- Each Neuron has soma, Dendrites,axon, synapses
- **Dendrites:** allow the cell to receive signals from number of neighboring neurons
- **Soma:** It aggregates information from dendrites
- If final information comming from dendrites is strong enough then signal send to axon



## Biological Neuron

- **Axon:** It tranfers signals
- **Axon Terminals (Synapses):**
- Axon terminals connect this neuron to other neurons
- Transfer information to other neurons

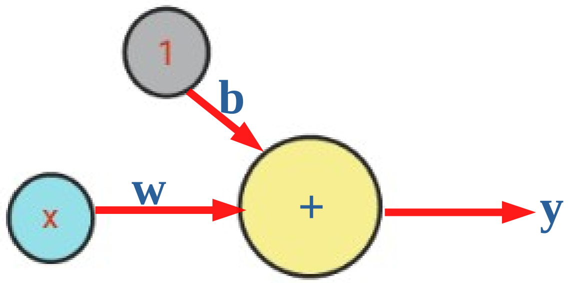


## Artificial Neuron

- Biological brains are capable of solving difficult problems
- But each neuron is only responsible for solving a very small part of the problem.
- Similarly, a neural network is made up of units(Artificial Neurons) that work together to produce a desired result
- Each neuron individually performs only a simple computation.

## Artificial Neuron(Linear Unit)

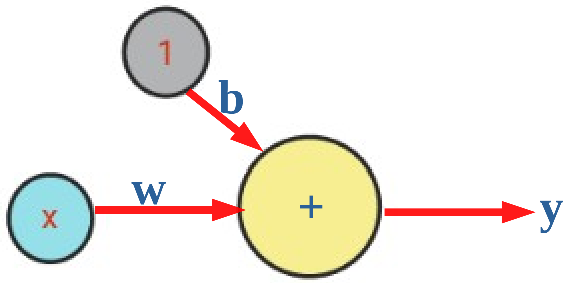
- The input is x.
- Its connection to the neuron has a weight which is w.
- Whenever a value flows through a connection, you multiply the value by the connection's weight.
- For the input x, what reaches the neuron is w * x.
- A neural network "learns" by modifying its weights.



**Source : https://www.kaggle.com/learn/intro-to-deep-learning**

# Artificial Neuron(Linear Unit)

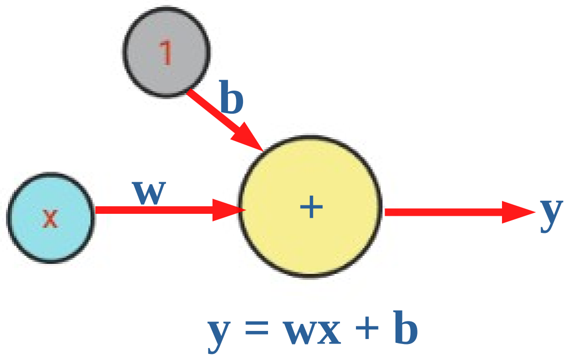
- The **b** is a special kind of **weight** we call the **bias**.
- The bias doesn't have any input data associated with it;
- We put a 1 in the diagram so that the value that reaches the neuron is just b (since 1 * b = b).
- The bias enables the neuron to modify the output independently of its inputs.



Source : https://www.kaggle.com/learn/intro-to-deep-learning

## Artificial Neuron(Linear Unit)

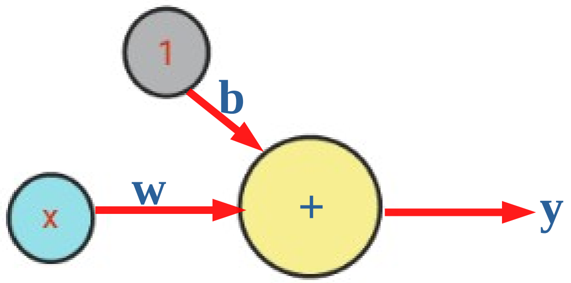
- The **y** is the value the neuron **outputs**.
- To get the output, the neuron **sums** up all the **values** it receives through its connections.
- This neuron's output is $y = w * x + b$.
- Bias value allows you to shift the output function to the left or right
- Artificial neuron is a mathematical function



Source : https://www.kaggle.com/learn/intro-to-deep-learning

## Artificial Neuron(Linear Unit)

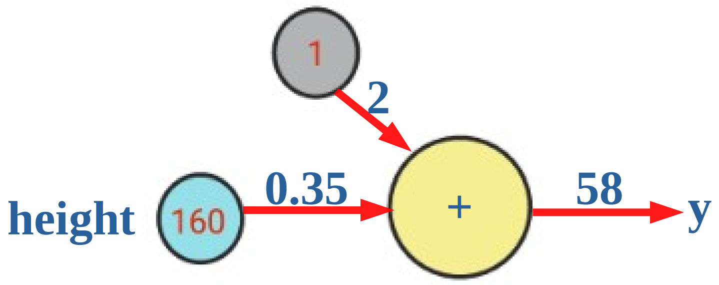
- The connections of the biological neuron are modeled as weights
- A positive weight reflects an excitatory connection
- Negative values mean inhibitory connections
- All inputs are modified by a weight and summed
- This activity is referred to as a linear combination



Source : https://www.kaggle.com/learn/intro-to-deep-learning

# Linear Unit as a model

- If we train this linear model with height(cm) as input and weight(kg) as output for some dataset
- If we find the bias is b=2 and the weight is w=0.35.
- We could estimate the weight of a person with 160 cm
- Weight would be 58kg

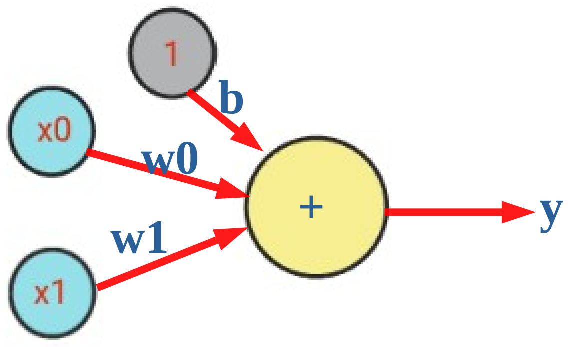


$$y = wx + b = 160 * 0.35 + 2 = 58$$

Source : https://www.kaggle.com/learn/intro-to-deep-learning

## Multiple Inputs(Linear Unit)

- Supose if dataset has one more feature gender
- We can just add one more input connection to the neuron
- To find the output, we would multiply each input to its connection weight and then add them all together.

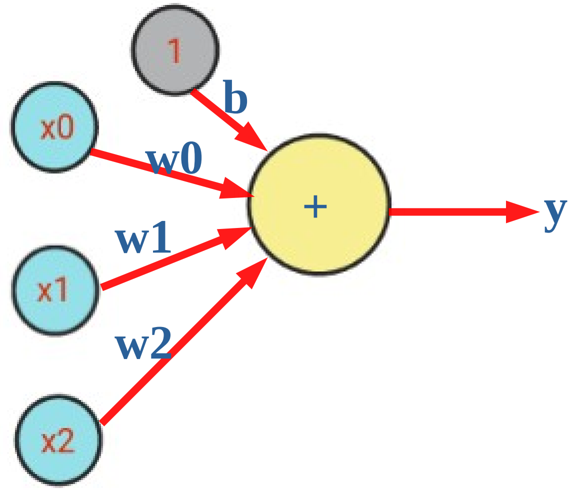


y = w0x0 + w1x1 + b

Source : https://www.kaggle.com/learn/intro-to-deep-learning

## Multiple Inputs(Linear Unit)

- If we have three inputs height, gender, country
- We can just add more input connections to the neuron, one for each additional feature.
- To find the output, we would multiply each input to its connection weight and then add them all together.



y = w0x0 + w1x1 +w2x2+ b

Source : https://www.kaggle.com/learn/intro-to-deep-learning

## Simple Linear Regression with Linear Units

## Linear classifier

- Classification is to use an object's characteristics to identify which class (or group) it belongs to.
- A linear classifier achieves this by making a classification decision based on the value of a linear combination of the features
- a classification algorithm that makes its predictions based on a linear predictor function.
- An example of the usage of a linear predictor function is in linear regression

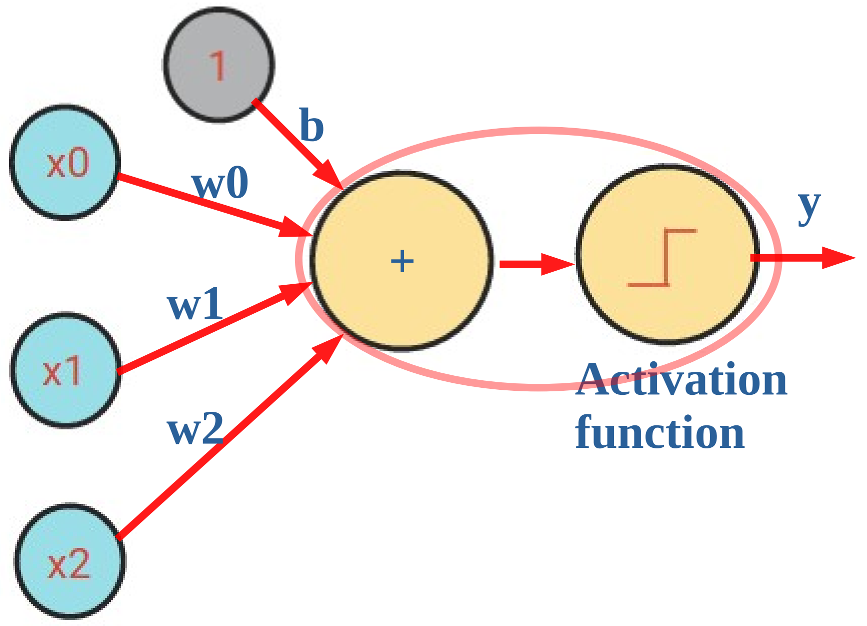
## Perceptron

- Perceptrons were developed in the 1958 by the scientist **Frank Rosenblatt**, inspired by earlier work by Warren **McCulloch** and Walter **Pitts**
- a perceptron is an artificial neuron using the Heaviside step function as the activation function.
- Perceptron is used for supervised learning of binary classifiers
- A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class.

**Reference: Michael A. Nielsen, "Neural Networks and Deep Learning" Determination Press 2015**

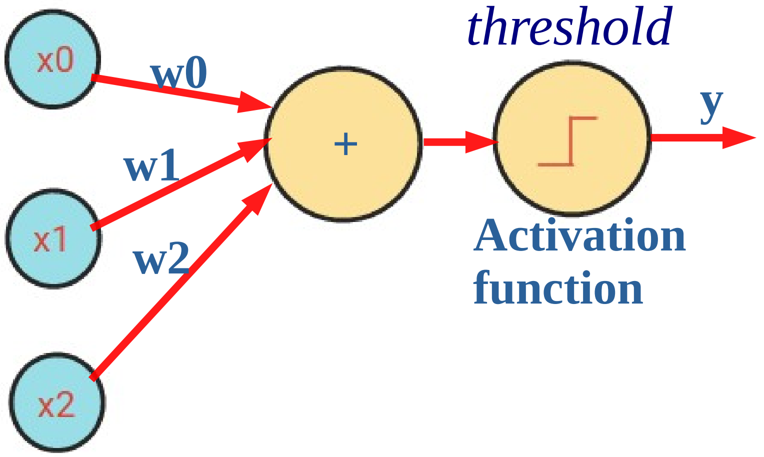
## Activation Function

- The weighted sum of inputs passed through an activation function
- An activation function **maps** the **weighted sum of inputs** to the **output of the neuron**(0,1).
- It is called an activation function because it **decide** whether neuron should be **activated or not**.



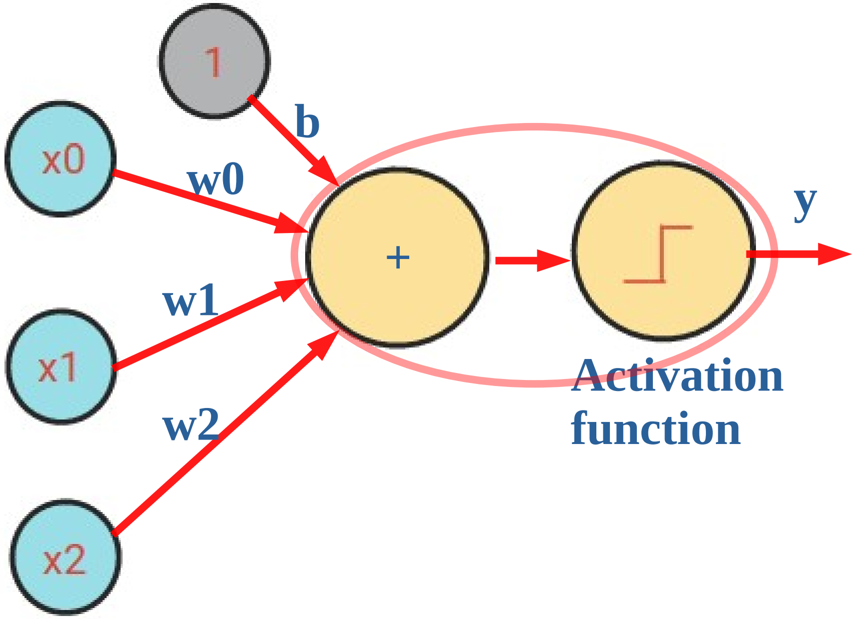
Reference: Michael A. Nielsen, “Neural Networks and Deep Learning” Determination Press 2015

## Perceptron



$$f(x)=\begin{cases}0 & \text{if } \sum_i w_i . x_i \leq threshold \\ 1 & \text{if } \sum_i w_i . x_i > threshold\end{cases}$$

$$f(x)=\begin{cases}0 & \text{if } w . x+b \leq 0 \\ 1 & \text{if } w . x+b > 0\end{cases}$$

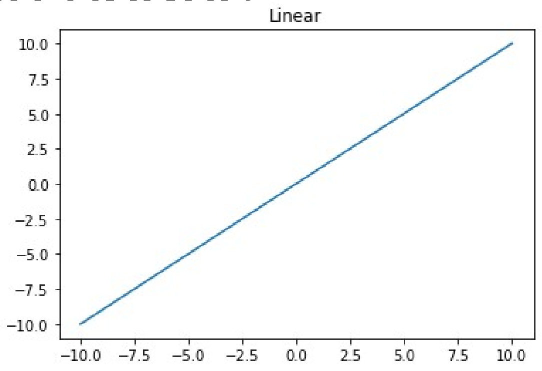


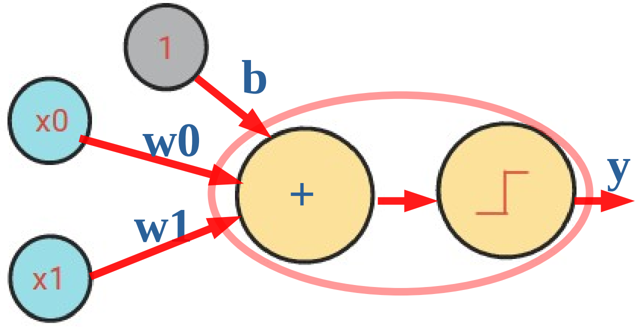
$$b \equiv - threshold$$

Reference: Michael A. Nielsen, "Neural Networks and Deep Learning" Determination Press 2015

# Perceptron

- If we have two inputs perception space of our model will be 2D plane
- Each input will be point on 2D plane
- Neuron can be imagined as stright line(sub space or hiper plane) in 2D perception space

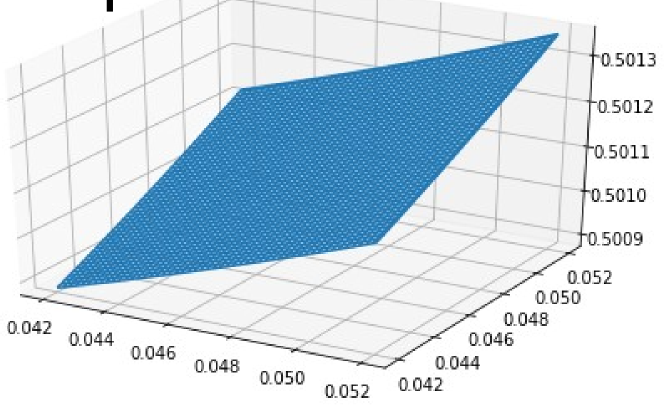


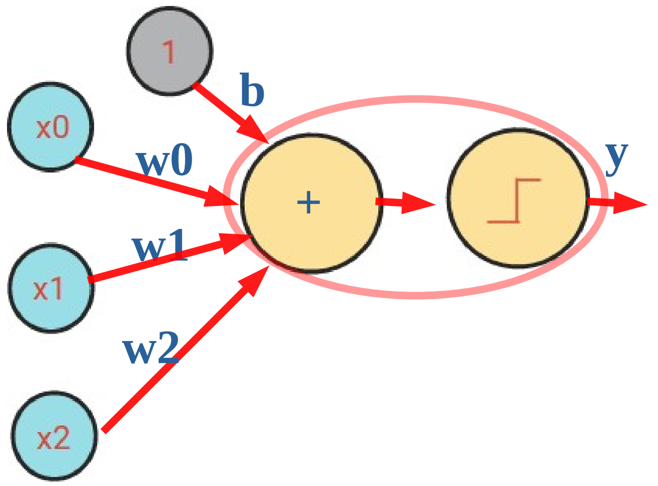


w0x0 + w1x1 + b = 0

## Perceptron

- If we have three inputs perception space of our model will be 3D plance
- Each input will be point on 3D plane
- Neuron can be imagined as 2D plane(hiper plane or sub space) in 3D pereption space.





w0x0 + w1x1 +w2x2+ b = 0

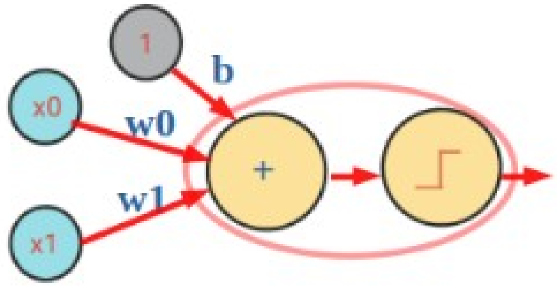
# Learning logic function OR

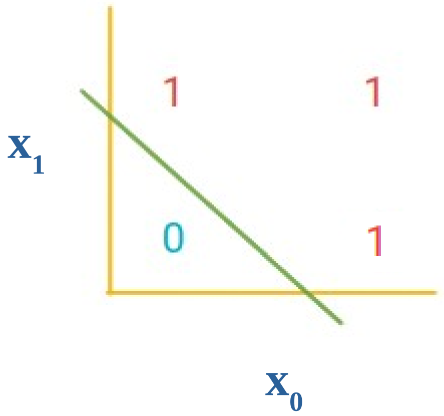
- If we consider the input ($x_0$,$x_1$ ) as a point on a plane
- The perceptron actually tells us to which region on the plane this point belongs.
- Perceptron equation will be

| $x_0$ | $x_1$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

$$w_0x_0 + w_1x_1 + b = 0$$

$$x_1 = -\frac{w_0}{w_1}x_0 - \frac{b}{w_1}$$

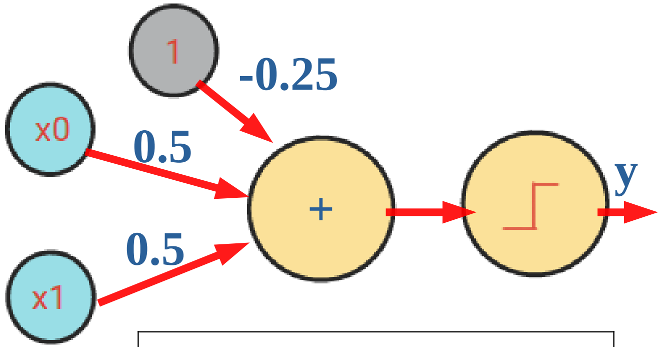




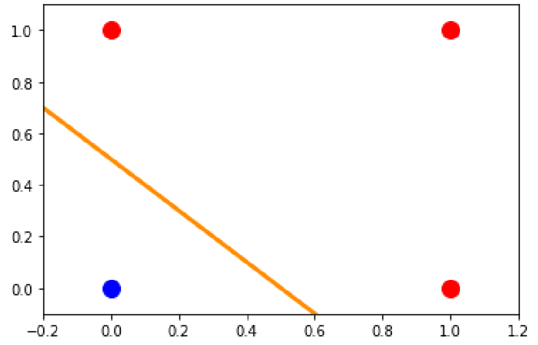
Reference: Michael A. Nielsen, “Neural Networks and Deep Learning” Determination Press 2015

# Learning logic function OR

- $w_0x_0 + w_1x_1 + b <= 0$ and $w_0x_0 + w_1x_1 + b > 0$ are the two regions on the plane Separated by the line $w_0x_0 + w_1x_1 + b = 0$



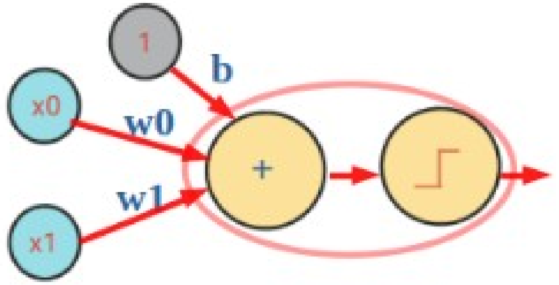
$$0.5*0+0.5*0-0.25=-0.25<0=0(C_0)$$
$$0.5*0+0.5*1-0.25=0.25>0=1(C_1)$$
$$0.5*1+0.5*0-0.25=0.25>0=1(C_1)$$
$$0.5*1+0.5*1-0.25=0.75>0=1(C_1)$$



Reference: Michael A. Nielsen, "Neural Networks and Deep Learning" Determination Press 2015

# Learning logic function XOR

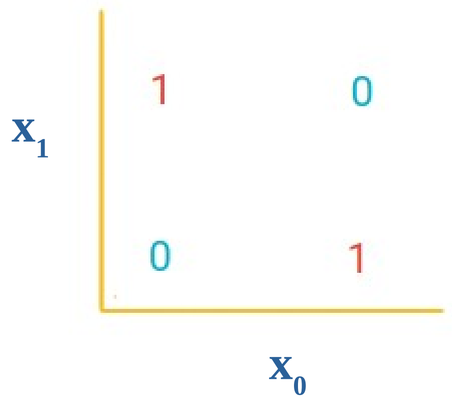
- If we consider the input $(x_0, x_1)$ as a point on a plane



| $x_0$ | $x_1$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- Perceptron equation will be

$$w_0 x_0 + w_1 x_1 + b = 0$$



Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.

## Single Neuron

- Each neuron individually performs only a simple computation.
- Single layer perceptrons are only capable of learning linearly separable patterns.
- a linear perceptron cannot be a universal classifier
- but that a network with a non polynomial activation function with one hidden layer can.

## Network with Multiple Neurons

# Network with Multiple Neurons

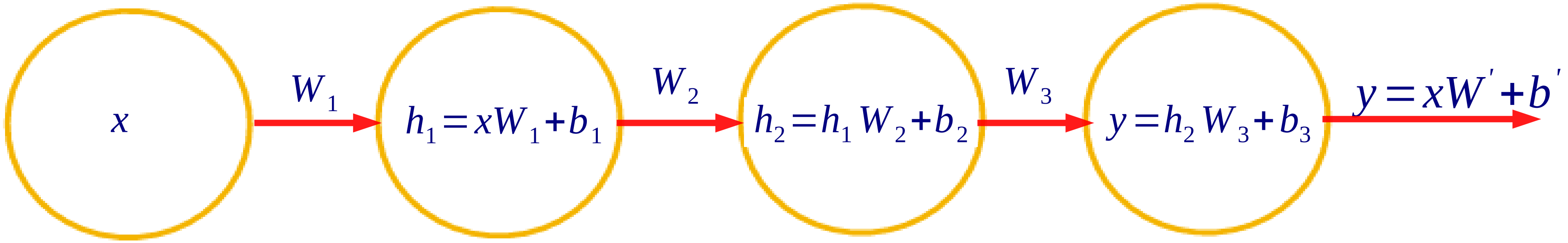
- Neural network with linear activation and any number of hidden layers is equivalent to just a linear neural neural network with no hidden layer.

$$\begin{tikzcd} x \arrow[r, "W_1"] & h_1=xW_1+b_1 \arrow[r, "W_2"] & h_2=h_1W_2+b_2 \arrow[r, "W_3"] & y=h_2W_3+b_3 \end{tikzcd}$$

$$y=h_2W_3+b_3$$
$$y=(h_1W_2+b_2)W_3+b_3$$
$$y=h_1W_2W_3+b_2W_3+b_3$$
$$y=(xW_1+b_1)W_2W_3+b_2W_3+b_3=xW_1W_2W_3+b_1W_2W_3+b_2W_3+b_3$$
$$y=xW'+b'$$

## Network with Multiple Neurons

- Combination of several linear transformations can be replaced with one transformation.
- Combination of several bias term is just a single bias out come is same even if we add linear activation



- Adding new layers does not increase the approximation power of linear neural network at all.
- We need non linear activation function to approximate non linear functions

## Network with Non Linear Activation function

## Non Linear Models

- The simplest way of modelling a nonlinear relationship is to transform the variables before estimating a regression model.

$$y = ae^{bx} U$$

$$\ln(y) = \ln(a) + bx + \ln(U)$$

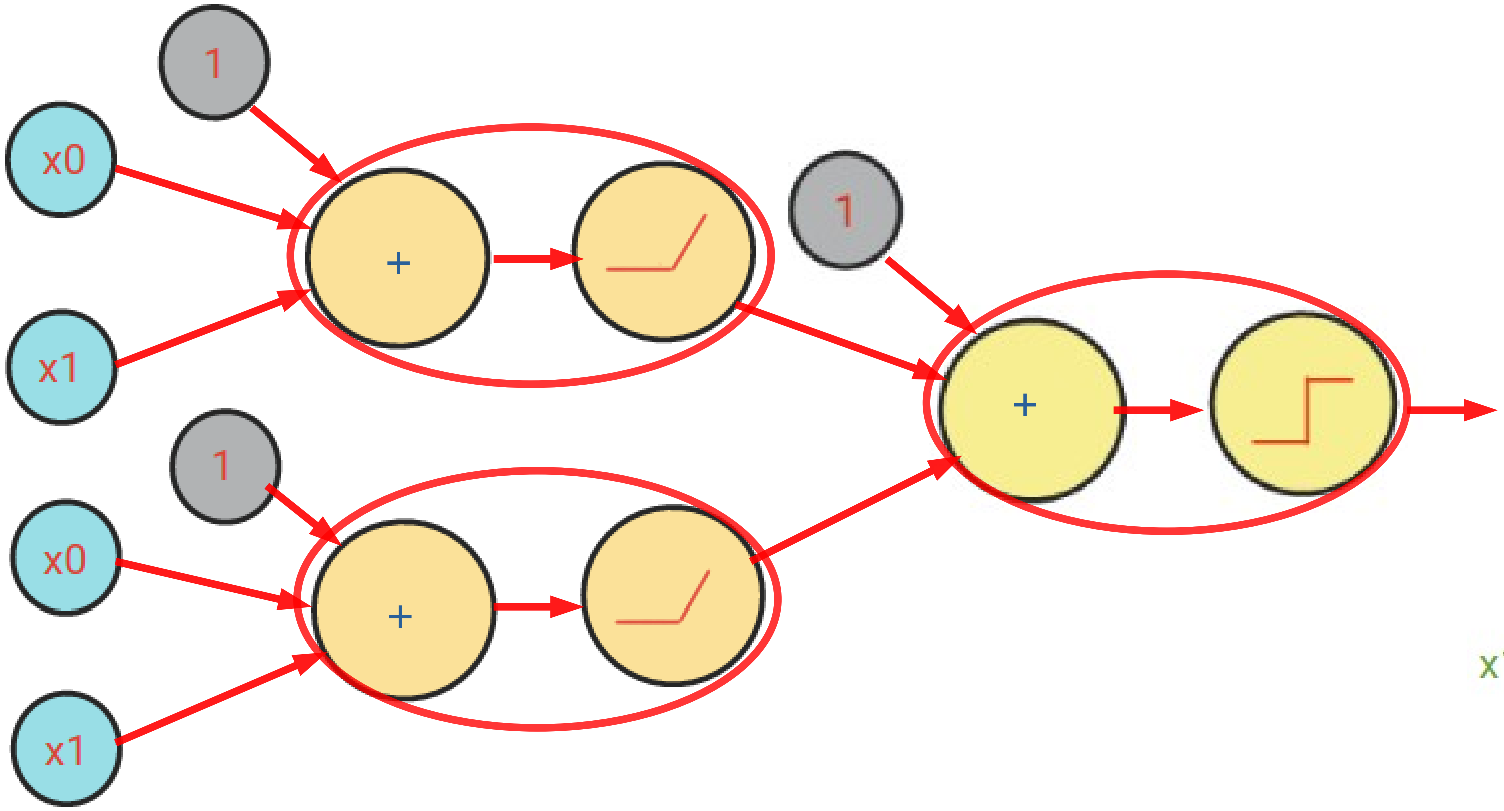
- This provides a non-linear functional form, the model is still linear in the parameters.

## Non Linear Models

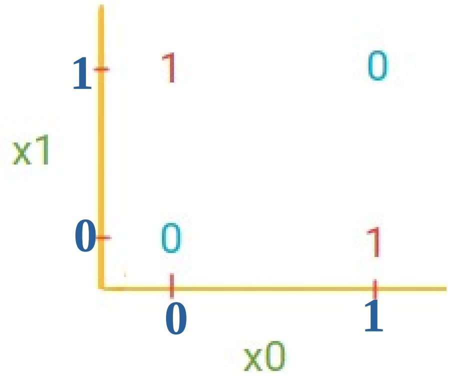
- Below regression equation is linear in parameters. But which can fit the curve

$$y = b + w_0 x_0 + w_1 x_0^2$$

# Learning logic function XOR

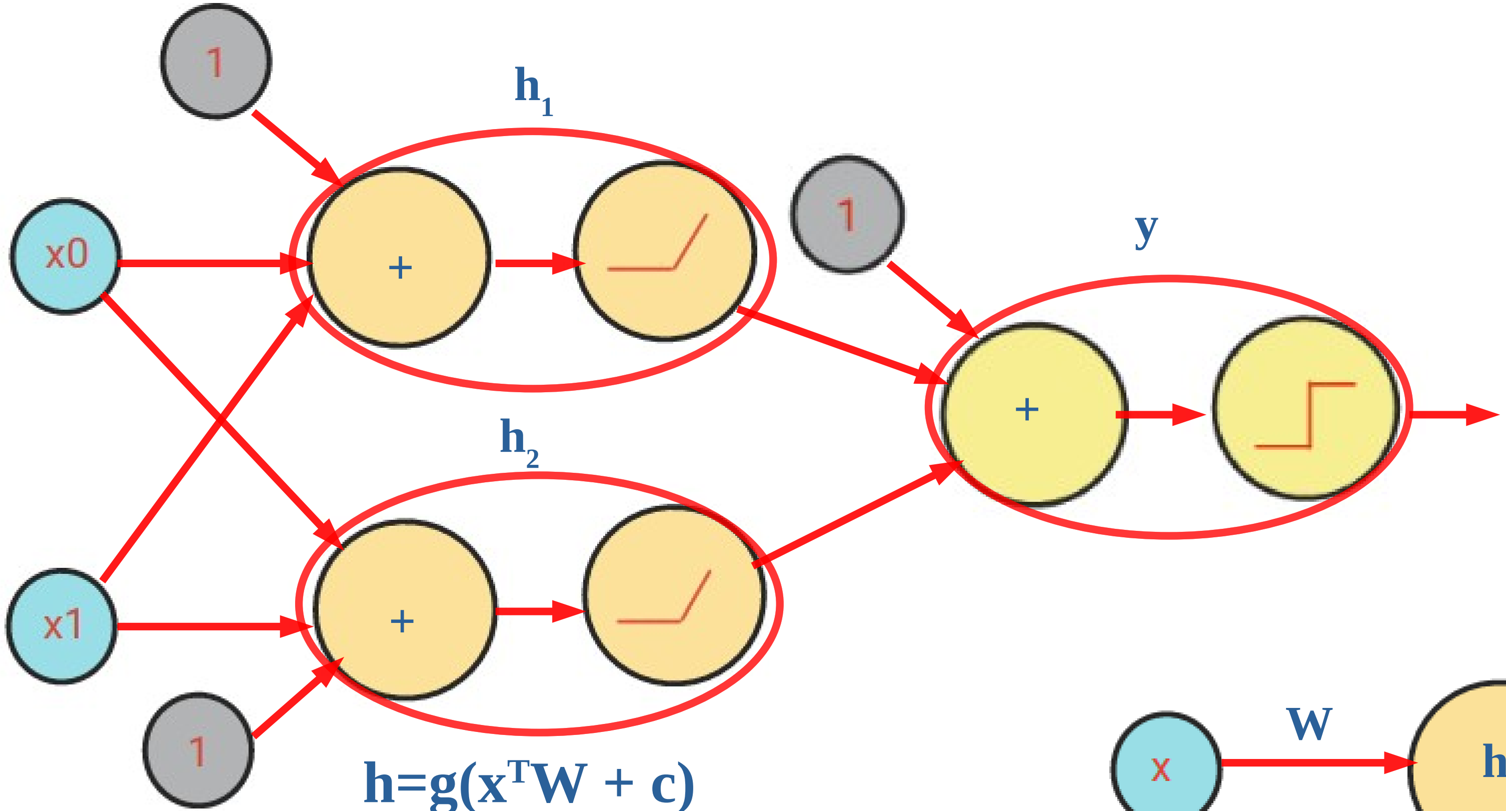


| $x_0$ | $x_1$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

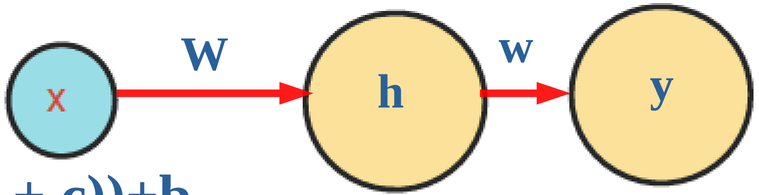


Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.

# Learning logic function XOR



| $x_0$ | $x_1$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



$h=g(x^TW + c)$

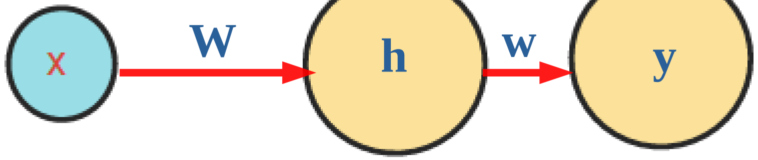
$f(x;W, c, w,b)=w^T \max(0,(W^Tx + c))+b$

Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.

## Learning logic function XOR

- Let $W=\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}$ $c=\begin{bmatrix}0\\-1\end{bmatrix}$ $w=\begin{bmatrix}1\\-2\end{bmatrix}$ and $b=0$

$$X=\begin{bmatrix}0 & 0\\0 & 1\\1 & 0\\1 & 1\end{bmatrix}$$



$f(x;W, c, w,b)=w^T \max(0,(W^T x + c))+b$

$$XW=\begin{bmatrix}0 & 0\\1 & 1\\1 & 1\\2 & 2\end{bmatrix} \quad XW+c=\begin{bmatrix}0 & -1\\1 & 0\\1 & 0\\2 & 1\end{bmatrix}$$

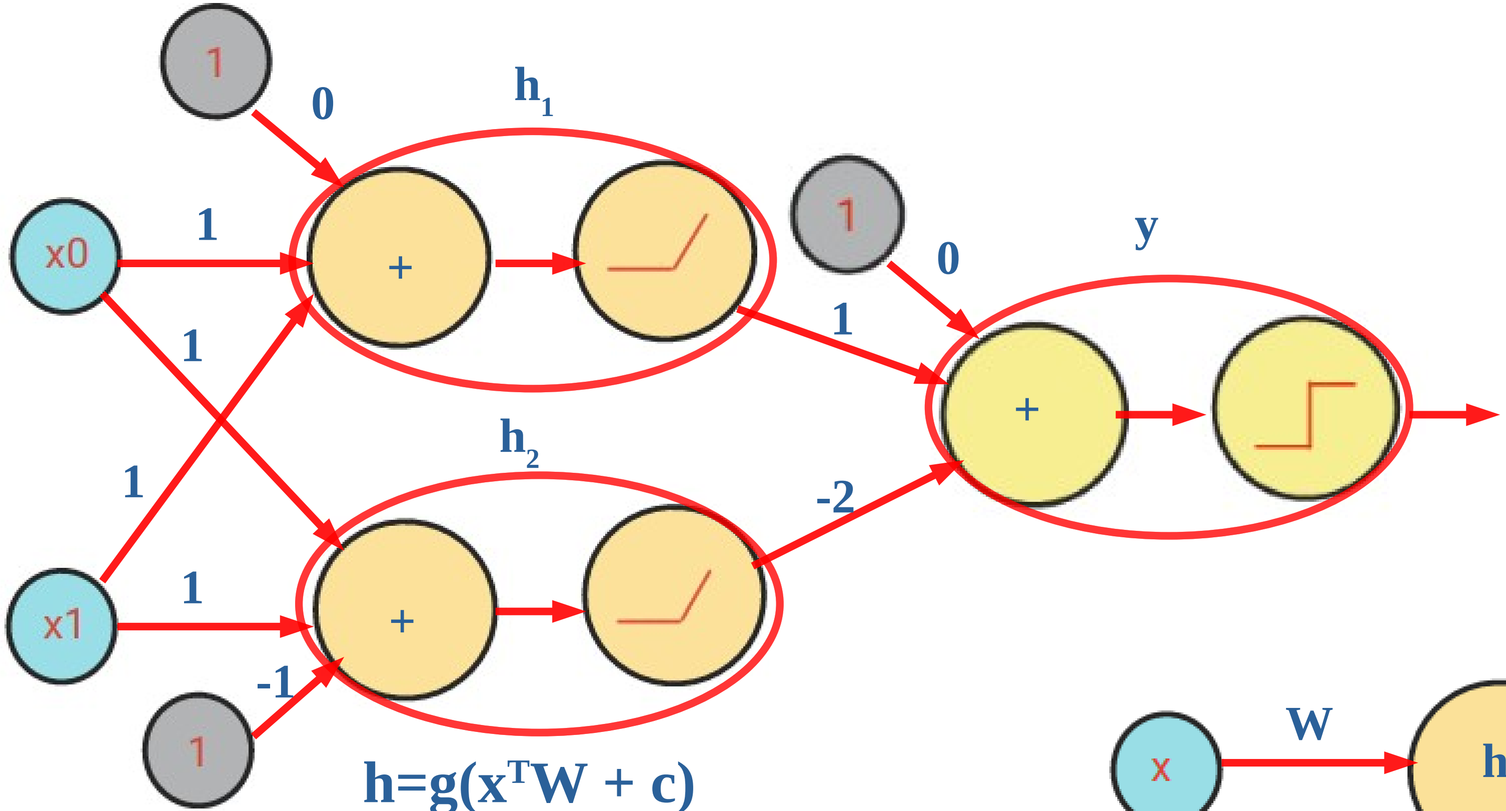
In this pace, all of the examples lie along aline with slope 1.

As we move along this line, the output needs to begin at 0, then rise to 1,then drop back down to 0.

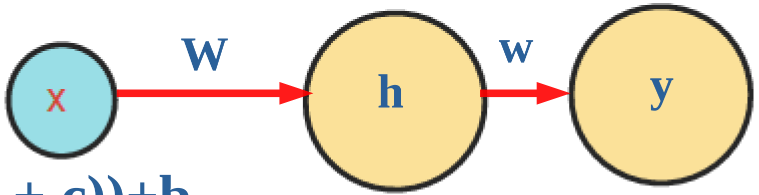
A linear model can not implement such a function

Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.

# Learning logic function XOR



| $x_0$ | $x_1$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



$$h=g(x^TW + c)$$

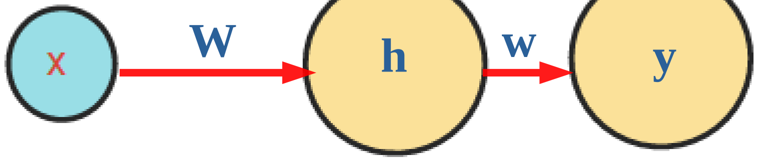
$$f(x;W, c, w,b)=w^T \max(0,(W^Tx + c))+b$$

Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.

## Learning logic function XOR

- Let $W=\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}$ $c=\begin{bmatrix}0\\-1\end{bmatrix}$ $w=\begin{bmatrix}1\\-2\end{bmatrix}$ and $b=0$

$$X=\begin{bmatrix}0 & 0\\0 & 1\\1 & 0\\1 & 1\end{bmatrix}$$



$f(x;W, c, w,b)=w^T \max(0,(W^Tx + c))+b$

$$XW=\begin{bmatrix}0 & 0\\1 & 1\\1 & 1\\2 & 2\end{bmatrix} \quad XW+c=\begin{bmatrix}0 & -1\\1 & 0\\1 & 0\\2 & 1\end{bmatrix} \quad h=\begin{bmatrix}0 & 0\\1 & 0\\1 & 0\\2 & 1\end{bmatrix}$$

**To finish computing the value of h for each example, we apply the rectified linear transformation x transformed into h. It has changed the relationship between the examples.**

**Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.**

## Learning logic function XOR

- Let $W=\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$ $c=\begin{bmatrix}0\\ -1\end{bmatrix}$ $w=\begin{bmatrix}1\\ -2\end{bmatrix}$ and $b=0$

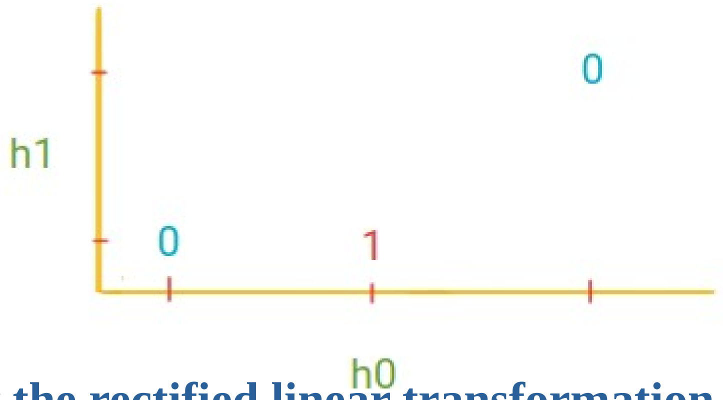
$$X=\begin{bmatrix}0 & 0\\ 0 & 1\\ 1 & 0\\ 1 & 1\end{bmatrix}$$

$$\begin{tikzcd} x \arrow[r, "W"] & h \arrow[r, "w"] & y \end{tikzcd}$$

$f(x;W, c, w,b)=w^T \max(0,(W^Tx + c))+b$

$$XW=\begin{bmatrix}0 & 0\\ 1 & 1\\ 1 & 1\\ 2 & 2\end{bmatrix}\quad XW+c=\begin{bmatrix}0 & -1\\ 1 & 0\\ 1 & 0\\ 2 & 1\end{bmatrix}\quad h=\begin{bmatrix}0 & 0\\ 1 & 0\\ 1 & 0\\ 2 & 1\end{bmatrix}$$

Learned h space



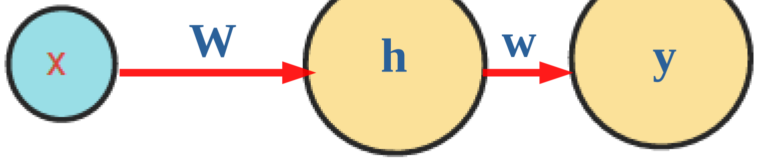
**To finish computing the value of h for each example, we apply the rectified linear transformation**

**Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.**

# Learning logic function XOR

- Let $W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ $c = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$ $w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$ and $b = 0$

$$X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$



$$f(x;W, c, w,b)=w^T \max(0,(W^T x + c))+b$$

$$XW = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} \quad XW + c = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} \quad h = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} \quad hw = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

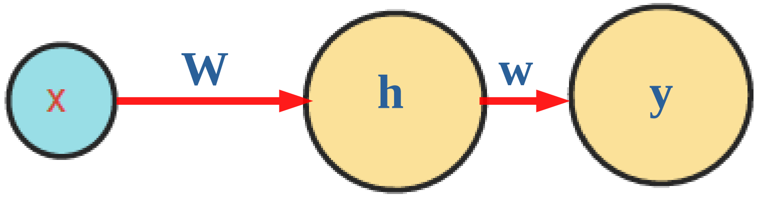
**Multiply h by the weight vector w**

**Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.**

# Learning logic function XOR

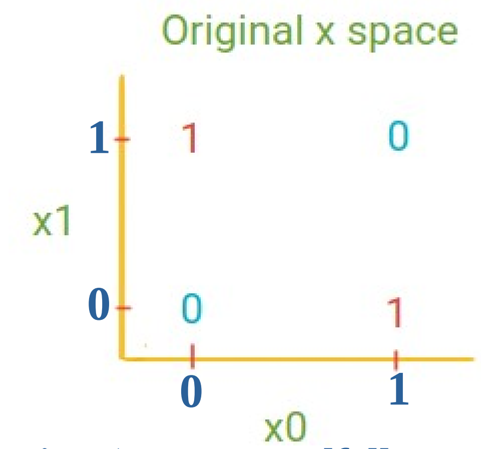
- Let $W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ $c = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$ $w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$ and $b = 0$

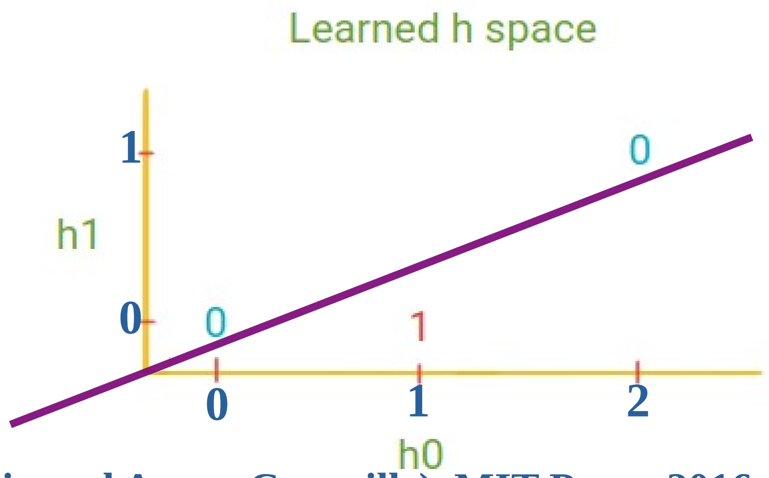
$$X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$



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

**$f(x;W, c, w,b)=w^T \max(0,(W^Tx + c))+b$**





**Reference: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.**

## Summary

- We used multiple layer neural network approach to sove XOR(non linear) problem.
- In this we followed two main steps.
- **First step:** we **transform the original** input data into new space using a nonlinear mapping.
- Once the data have been transformed into the new space,
- **Second step:** searches for a **linear separating hyperplane in the new space**.
- The hyperplane found in the new space corresponds to a nonlinear hyperplane in the original space.

## Summary

- You can assume it like the kernel trick you studied in SVM
- SVM with kernel given by φ((a, b)) = (a, b, $a^2 + b^2$) and thus $K(x, y) = x \cdot y + ||x||^2 ||y||^2$
- The training points are mapped to a 3-dimensional space where a separating hyperplane can be easily found.