

Deep Learning & Neural Networks

Dr. Shahira Ahmed

DS710

Lecture 2





Faculty of Engineering
جامعة ميدوcean
الميدوcean

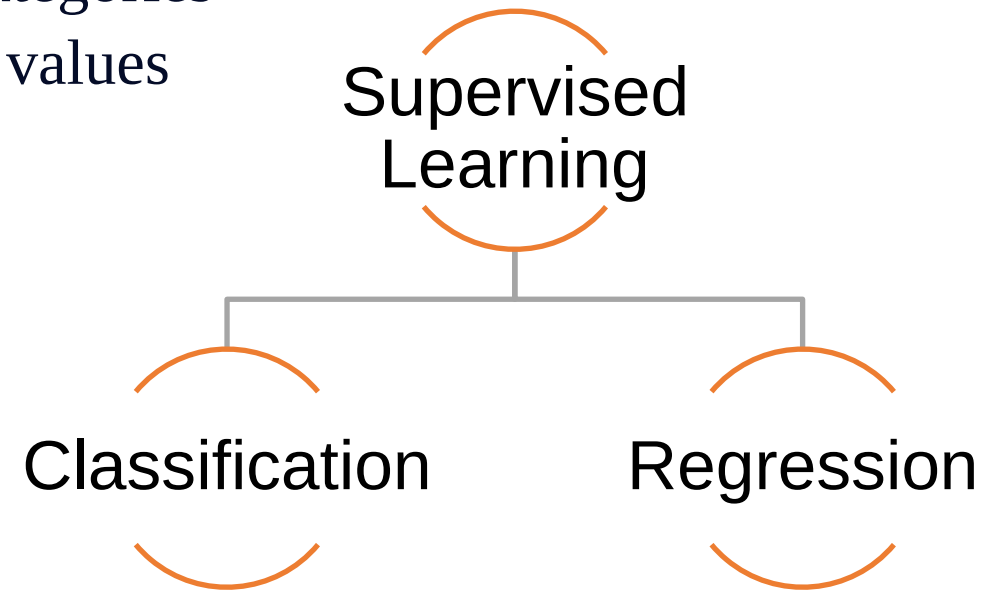
Deep Learning & Neural Networks

DS710

Lecture 2: Perceptron

- Linear Regression
- Logistic Regression
- Perceptron
- Perceptron as logic gate
- Artificial Neural Network
 - Shallow
 - Deep

- Classification and regression are machine learning tasks, but they differ in output.
 - Classification predicts discrete labels or categories
 - Regression predicts continuous numerical values



Linear Regression Example

- Predicting house price house prices based on features such as size, location, and number of bedrooms

Size	Price
10	100
20	200
30	150
40	300
50	500



- Linear Regression is a Supervised Learning method, where the predicted output will be continuous → Regression Task

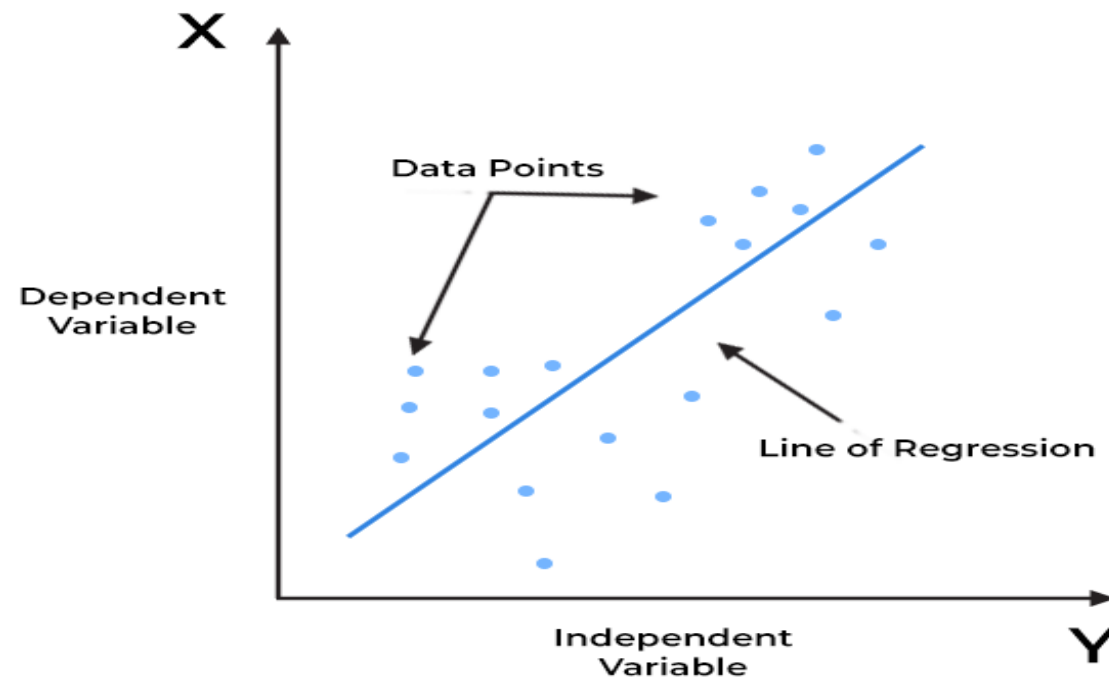
Linear Regression

- Linear Regression is a Supervised Learning method, where the predicted output will be continuous → Regression Task
- Imagine the task of trying to predict the price of a house given its size

Size	Price
10	100
20	200
30	150
40	300
50	500

Independent
Variable

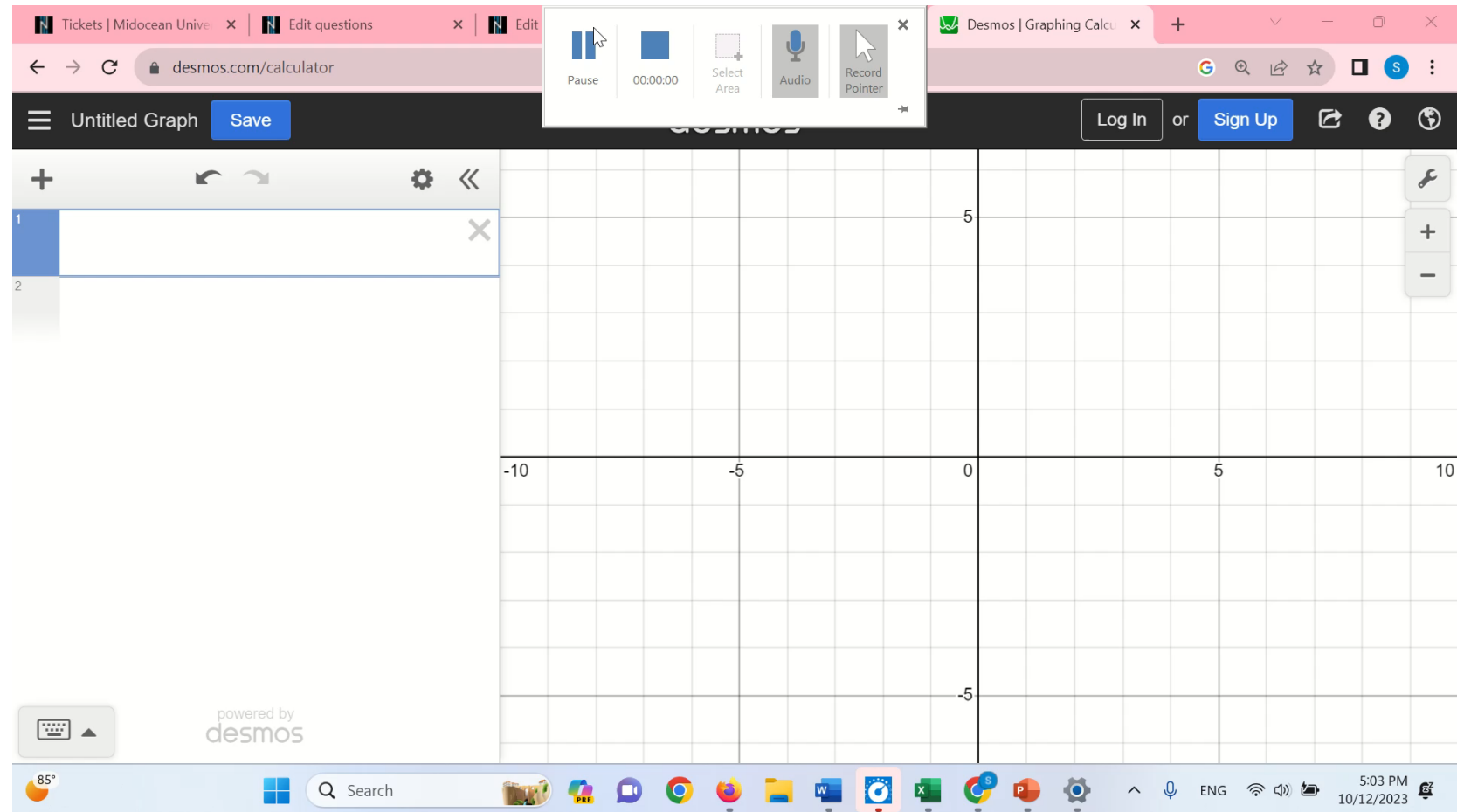
dependent
Variable



- Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

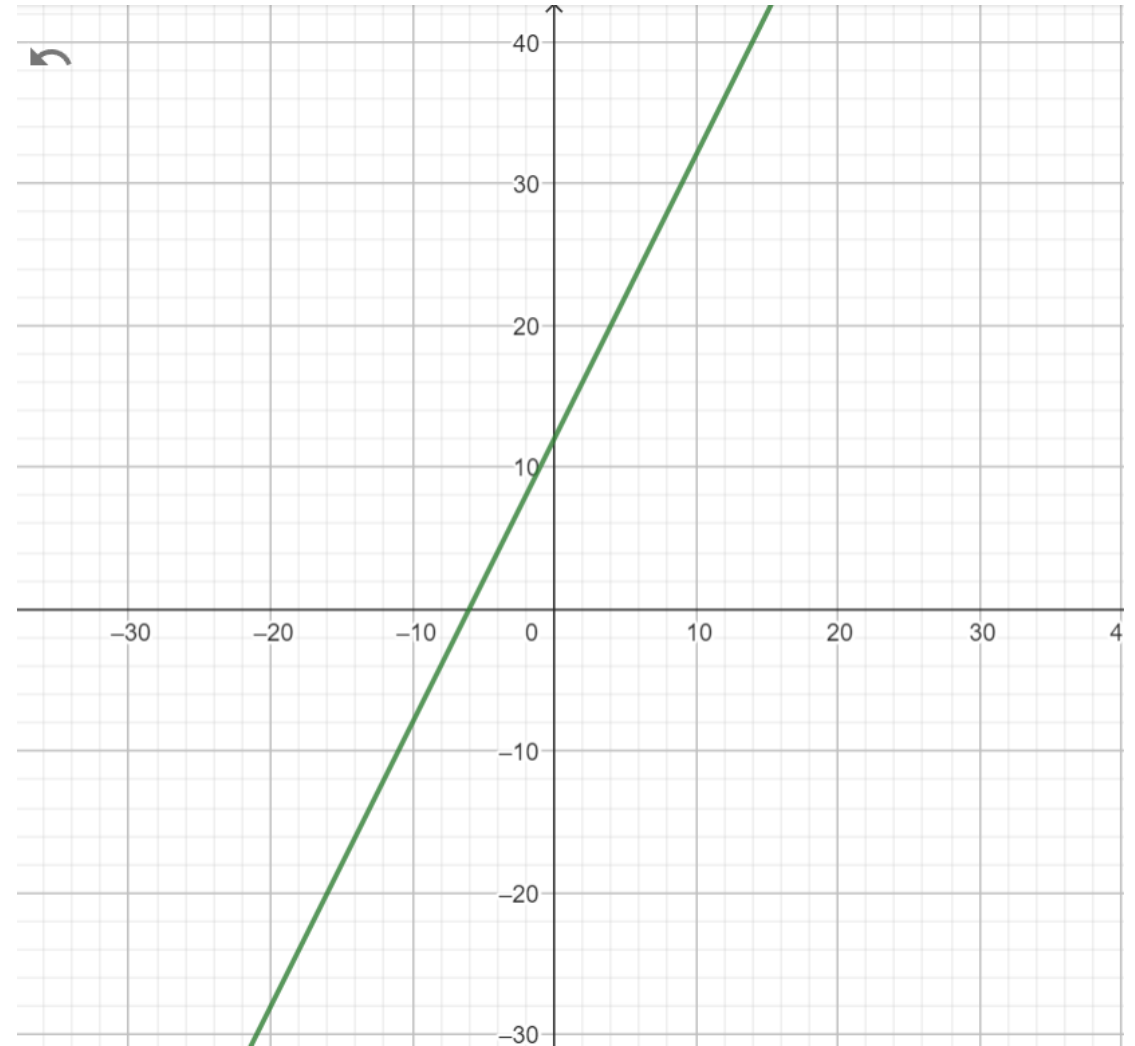
Moving a Line

- $y = w_1x + w_2$
- w_1 : Slope
- w_2 : y - intercept

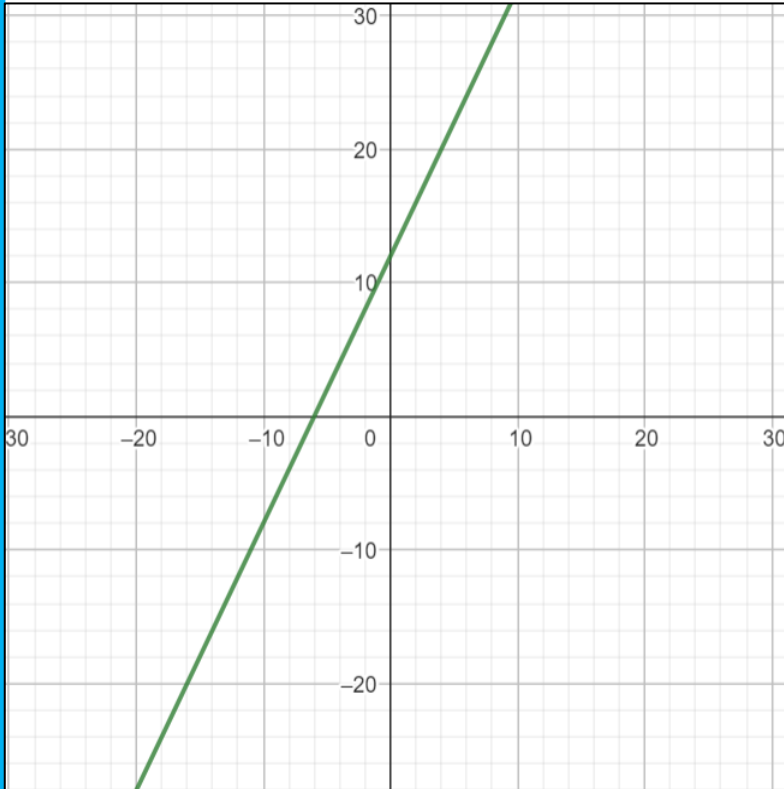


Moving a Line

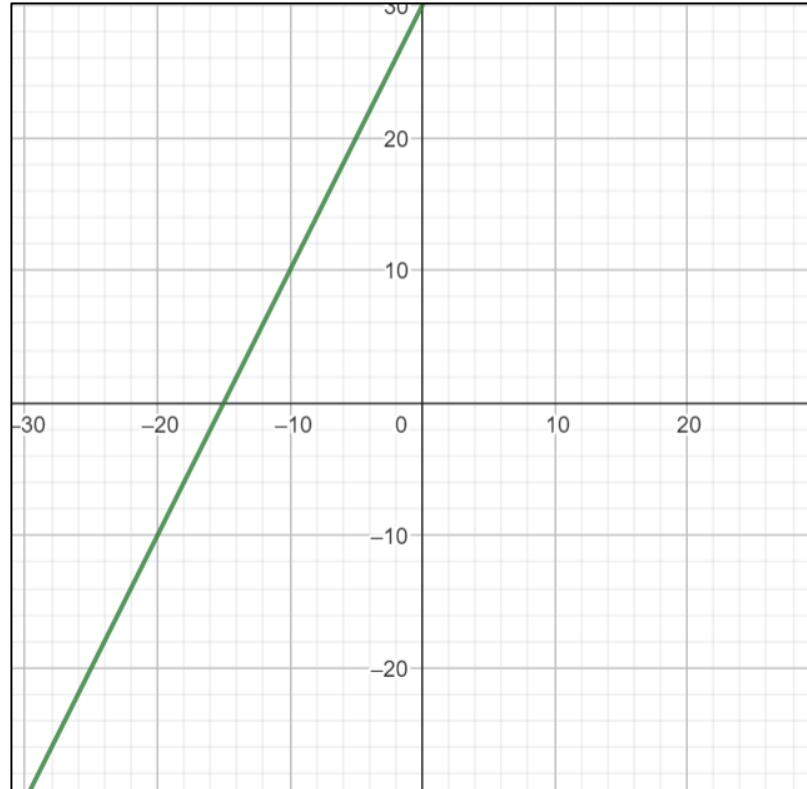
- $y = w_1x + w_2$
- w_1 : Slope
- w_2 : y - *intercept*
- e.g.: $2x+12$



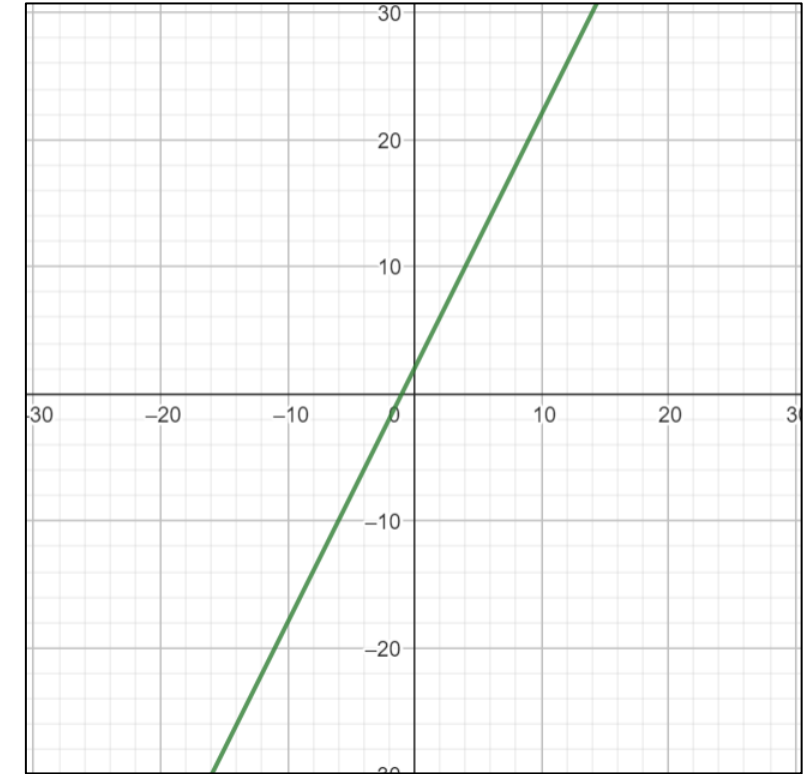
Moving a Line – changing y-intercept



$$f(x) = 2x + 12$$

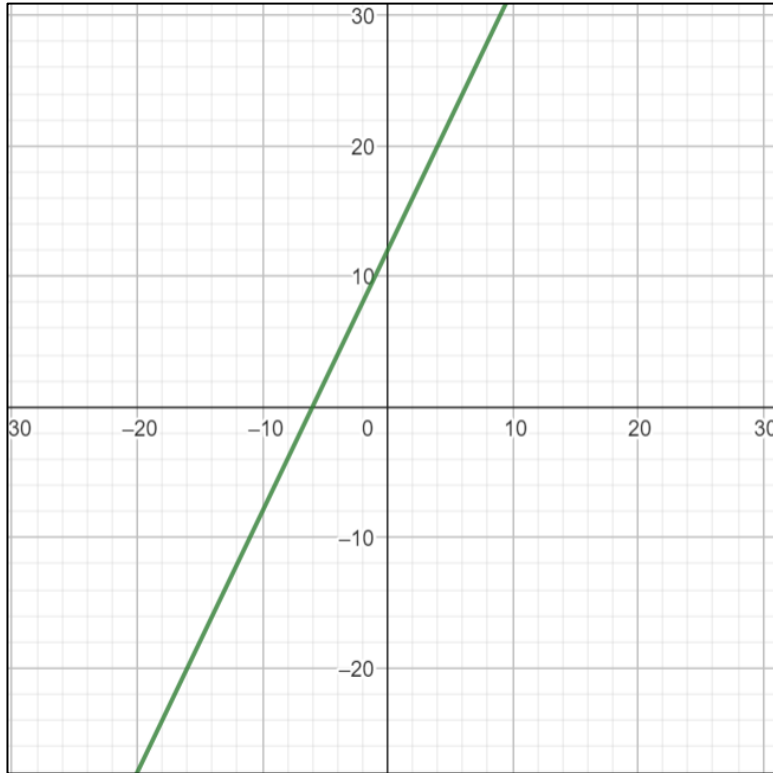


$$f(x) = 2x + 30$$

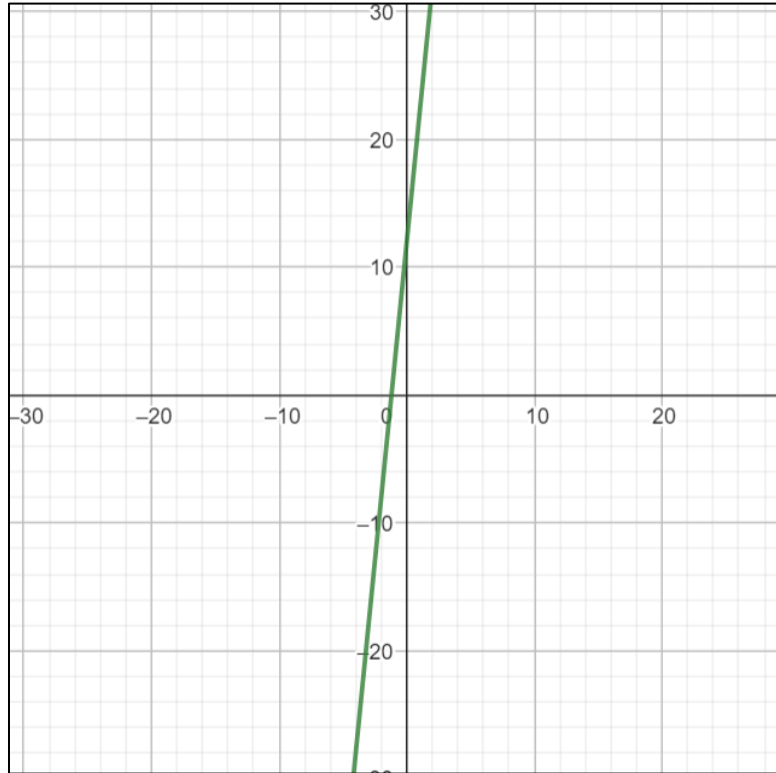


$$f(x) = 2x + 2$$

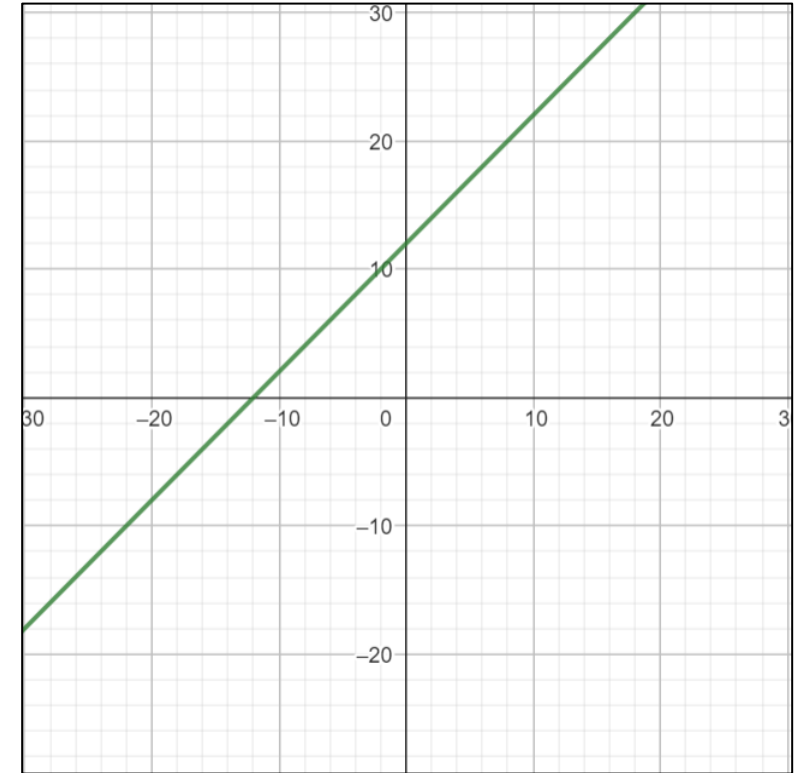
Moving a Line – changing slope



$$f(x) = 2x + 12$$

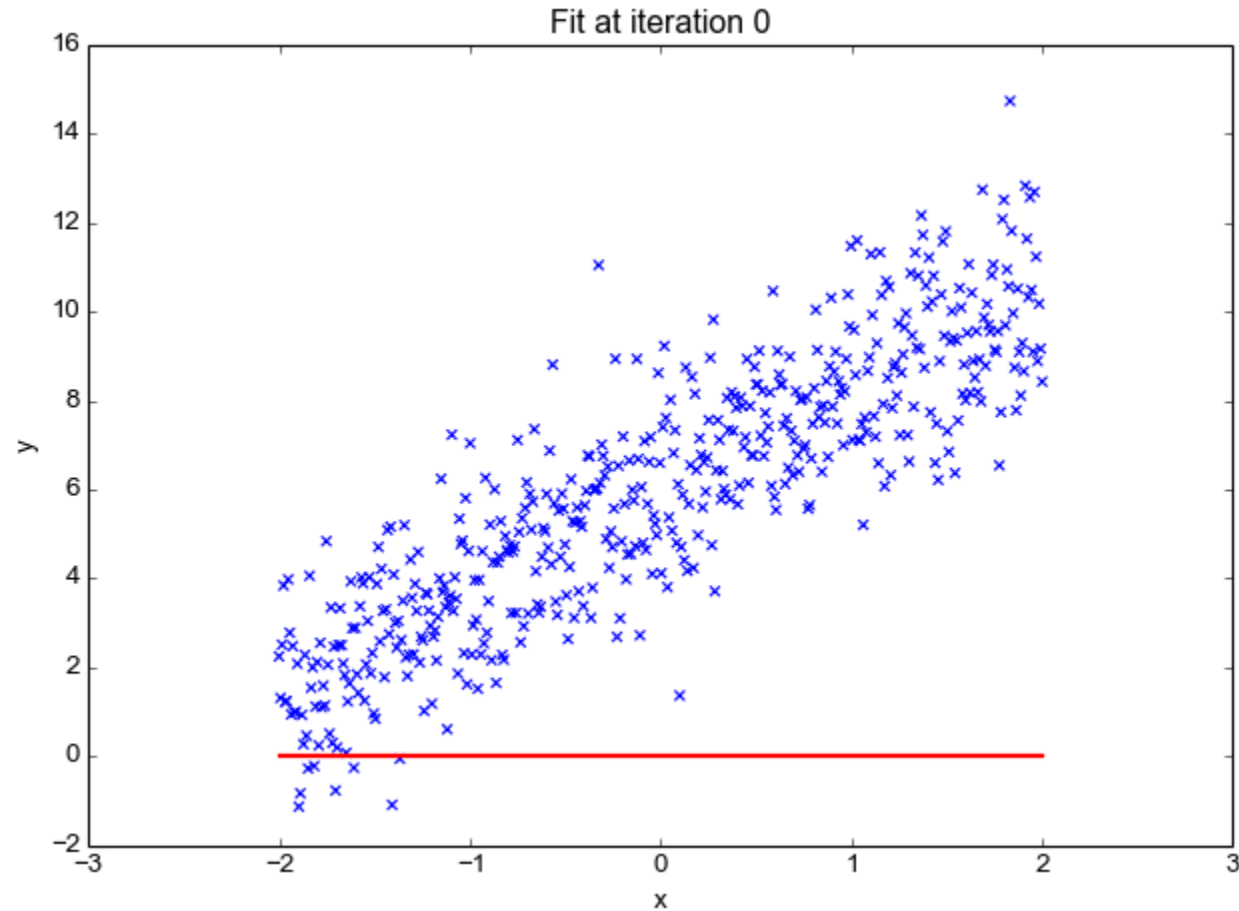


$$f(x) = 10x + 12$$



$$f(x) = x + 12$$

Linear Regression



[Linear Regression - PRIMO.ai](#)

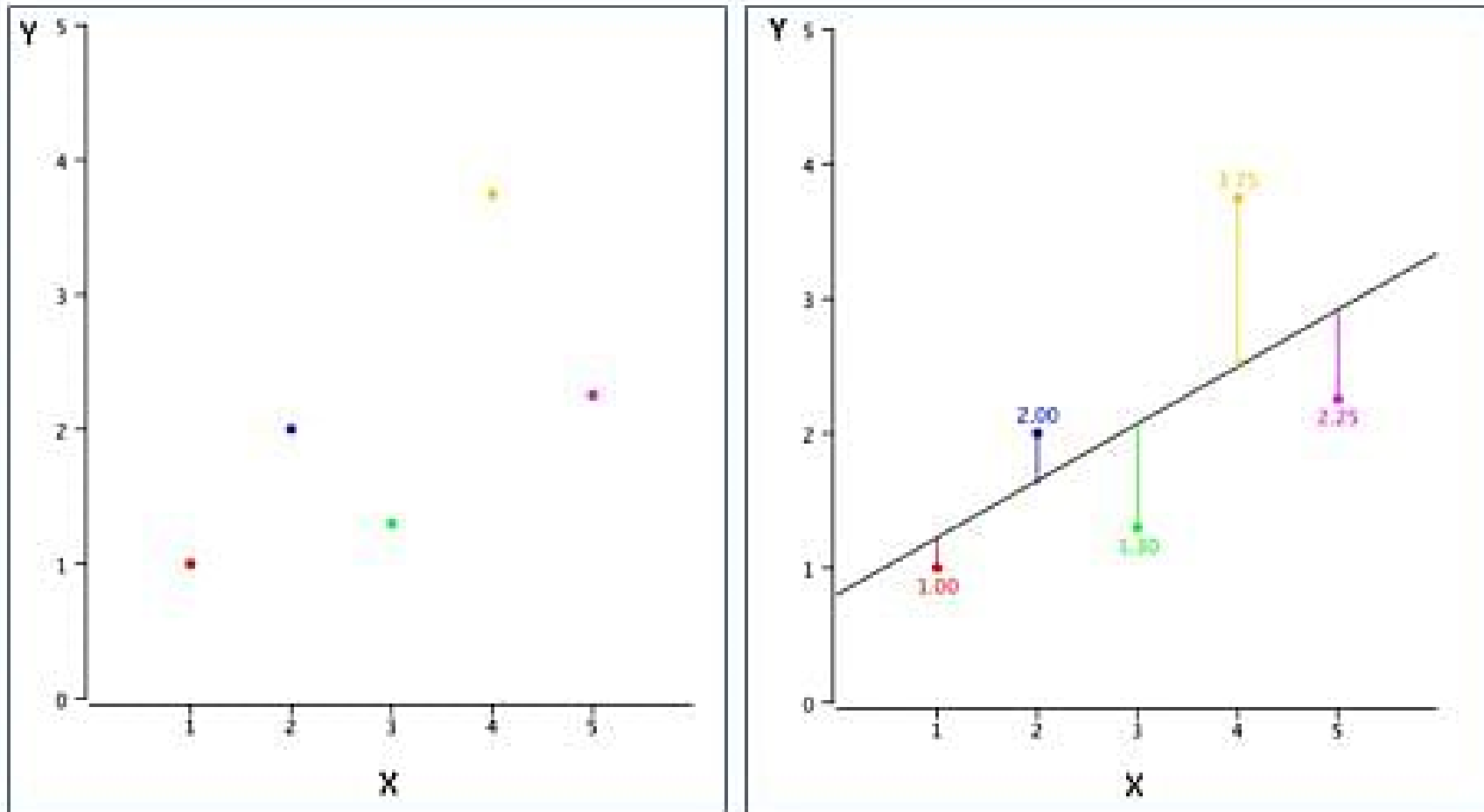
- The primary goal of linear regression is to find the "best-fit" line (or hyperplane in higher dimensions) that minimizes the difference between the predicted values and the actual observed values.
- The idea : trying to draw a line (called 'fitting') through an entire dataset of points. What is the optimum line?



- Size of the house =500 what is the price?
- $y = w_1x + w_2 \Rightarrow w_1=5, w_2=100$
- Solution
- $5*500+100=2600$

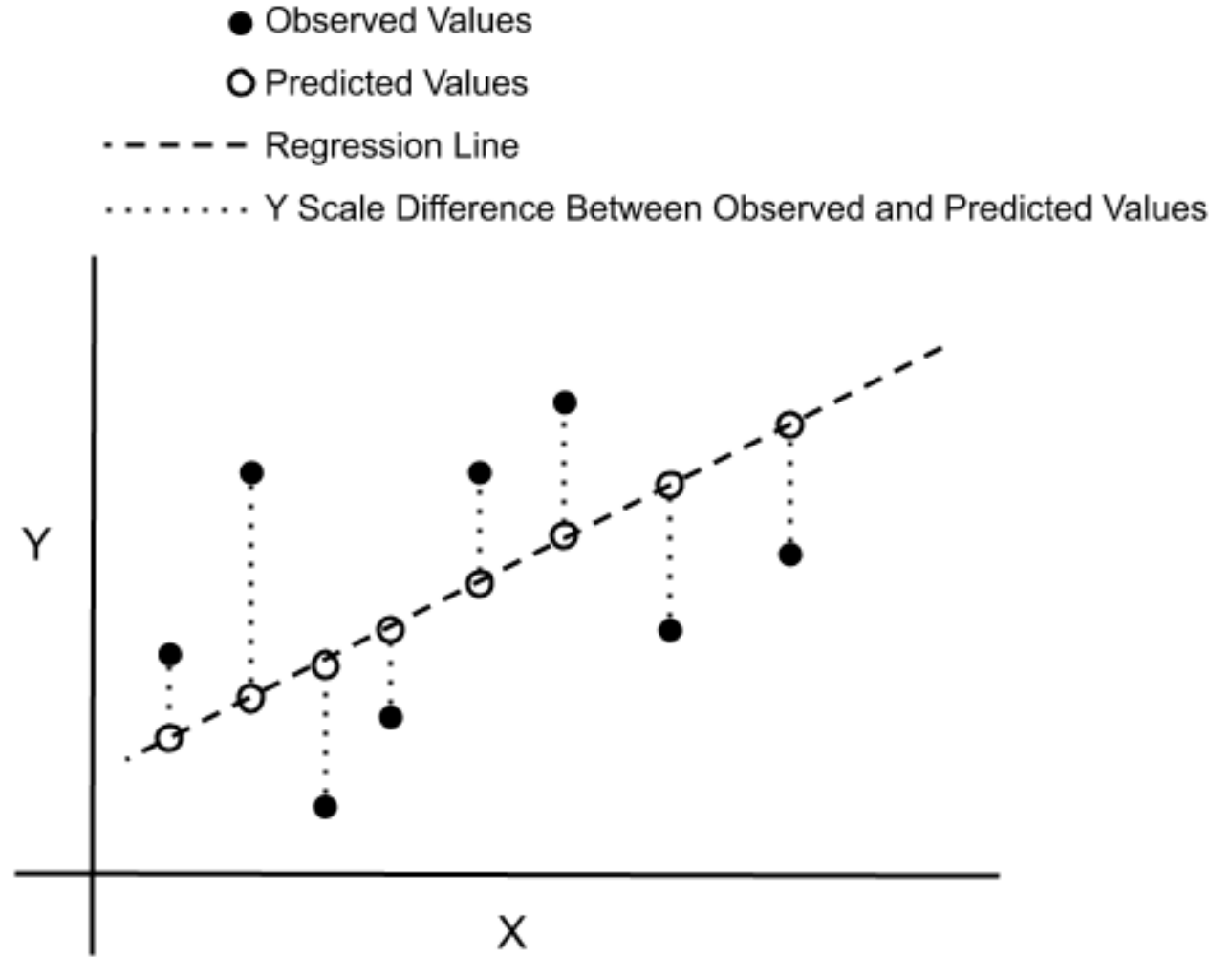
- Size of the house =500,no of bedroom =4, what is the price?
- $y = w_1x_1 + w_2x_2 + w_3 \Rightarrow w_1=5, w_2=10, w_3 =100$
- Solution
- $5*500+4*10+100$

Quiz: Predicting Price of a house



Mean Square Error

- Mean Squared Error (MSE) measures the average squared difference between estimated values and the actual value. MSE is:
- always a zero or positive value
- better for values closer to zero



<https://www.ml-science.com/mean-squared-error>

- Actual- Predicted

The diagram illustrates the calculation of error. It shows two terms: Y_i and $\hat{Y}_i$. Above Y_i is the text "real value" in green, and above $\hat{Y}_i$ is the text "predicted value" in red. Both terms are enclosed in boxes of the same color. A minus sign is placed between the two boxes. A horizontal bracket underneath the entire expression $Y_i - \hat{Y}_i$ is labeled "error" in grey.

- House 1: actual 120K, predicted 100K -> error 20K
- House 2: actual 60K, predicted 80K -> error -20K
- If you sum these up the error will be 0? incorrect

- $\text{Error} = (\text{Actual} - \text{Predicted})^2 = \underbrace{(\overset{\text{real value}}{\boxed{Y_i}} - \overset{\text{predicted value}}{\boxed{\hat{Y}_i}})^2}_{\text{squared error}}$

- House 1: actual 120K, predicted 100K -> error 20K → $(20)^2 = 400$
- House 2: actual 60K, predicted 80K -> error -20K → $(-20)^2 = 400$

Mean Square Error

- We usually compute the error over multiple samples (in our example - houses)

- Error = $\sum_{i=1}^n (\text{Actual} - \text{Predicted})^2 =$

The diagram illustrates the Mean Square Error formula with color-coded annotations:
- **number of samples**: A blue box around the variable n in the summation index.
- **real value**: A green box around the variable Y_i .
- **predicted value**: A red box around the variable $\hat{Y}_i$.
The formula is $\sum_{i=1}^n (Y_i - \hat{Y}_i)^2$. A bracket underneath the entire summation is labeled "sum of the errors of all samples".

- House 1: actual 120K, predicted 100K $\rightarrow$ error 20K $\rightarrow (20)^2 = 400$
- House 2: actual 60K, predicted 80K $\rightarrow$ error -20K $\rightarrow (-20)^2 = 400$

Mean Square Error

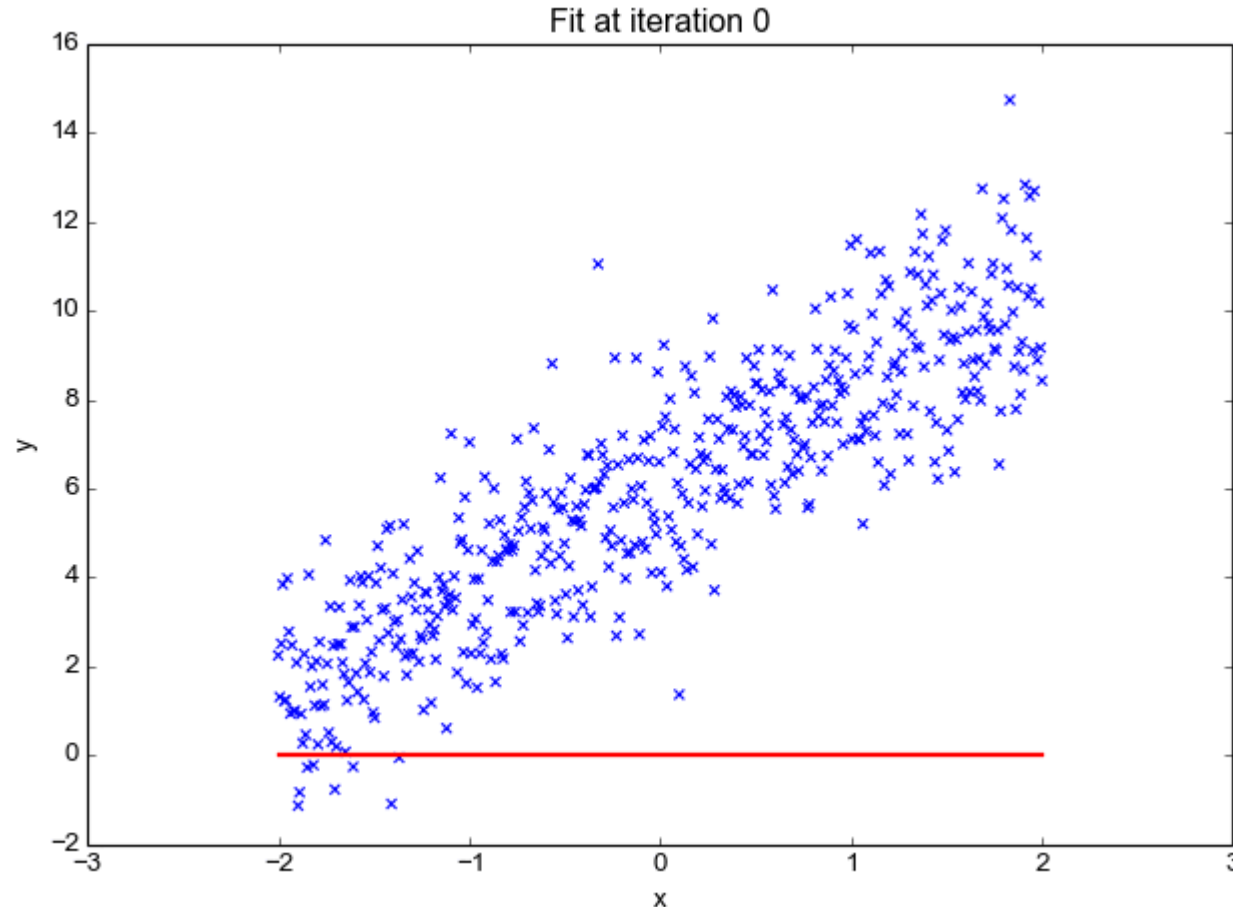
- if you want to compare the errors of batches of different sizes, you need to normalize for the number of samples - you take the average. For example, you may want to see which batch size produces a lower error
- $\text{Error} = \text{average}(\text{Actual} - \text{Predicted})^2 =$

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

number of samples n real value Y_i predicted value $\hat{Y}_i$

sum of the errors of all samples

- Assume relation between independent and dependent variables are Linear

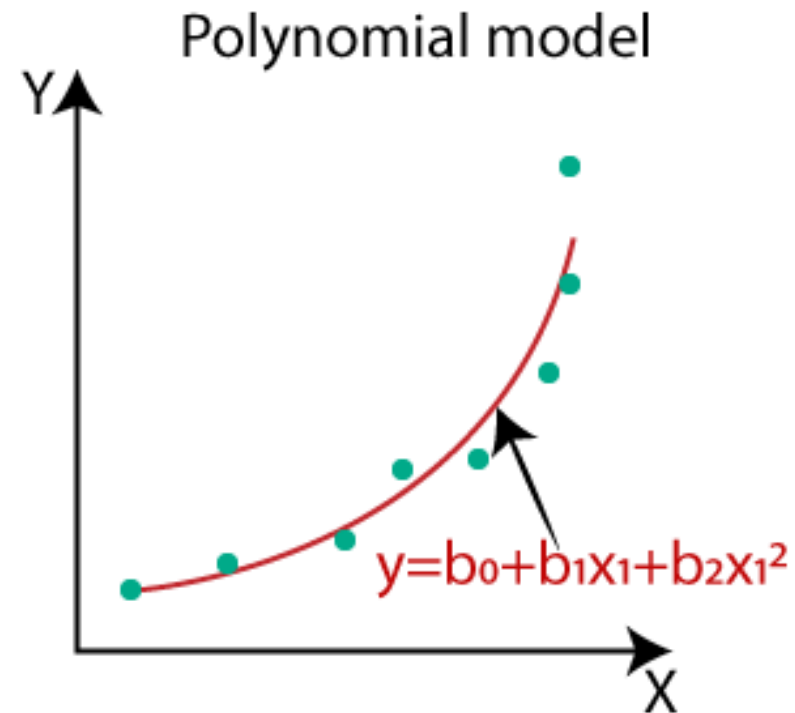
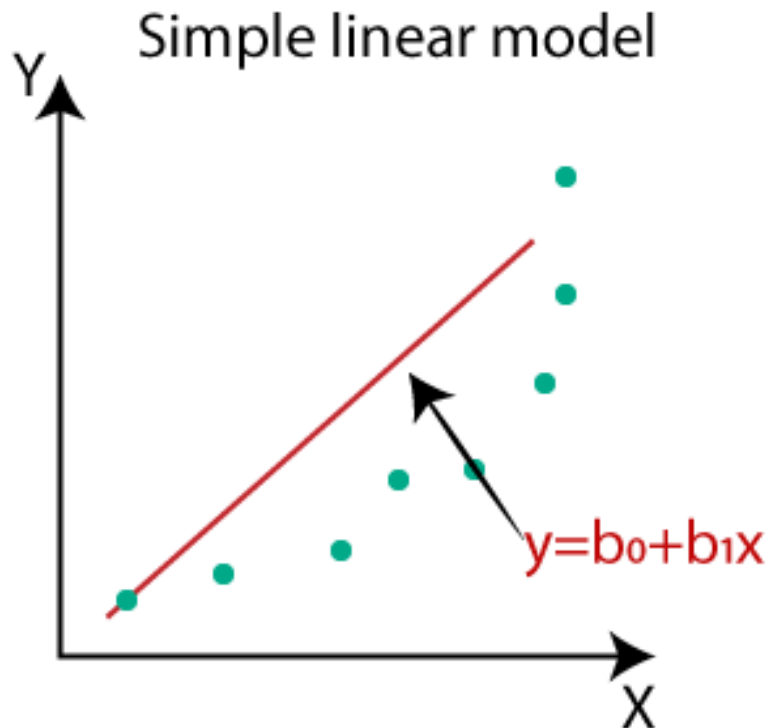


- First, draw a random line, and calculate the error
 - INITIALIZE random weights.
 - The error is some measure of how far the points are from the line. Calculate the average of these distances.
- Repeat
 - move the line around and see if we can decrease this error.
- until we get to the perfect line.

- Data
- Training (Fitting Line using gradient descent (MSE))
- Prediction $w_1x_1 + w_2x_2 + \dots + w_nx_n + b$

Polynomial Regression

- Used for **non-linear dataset**



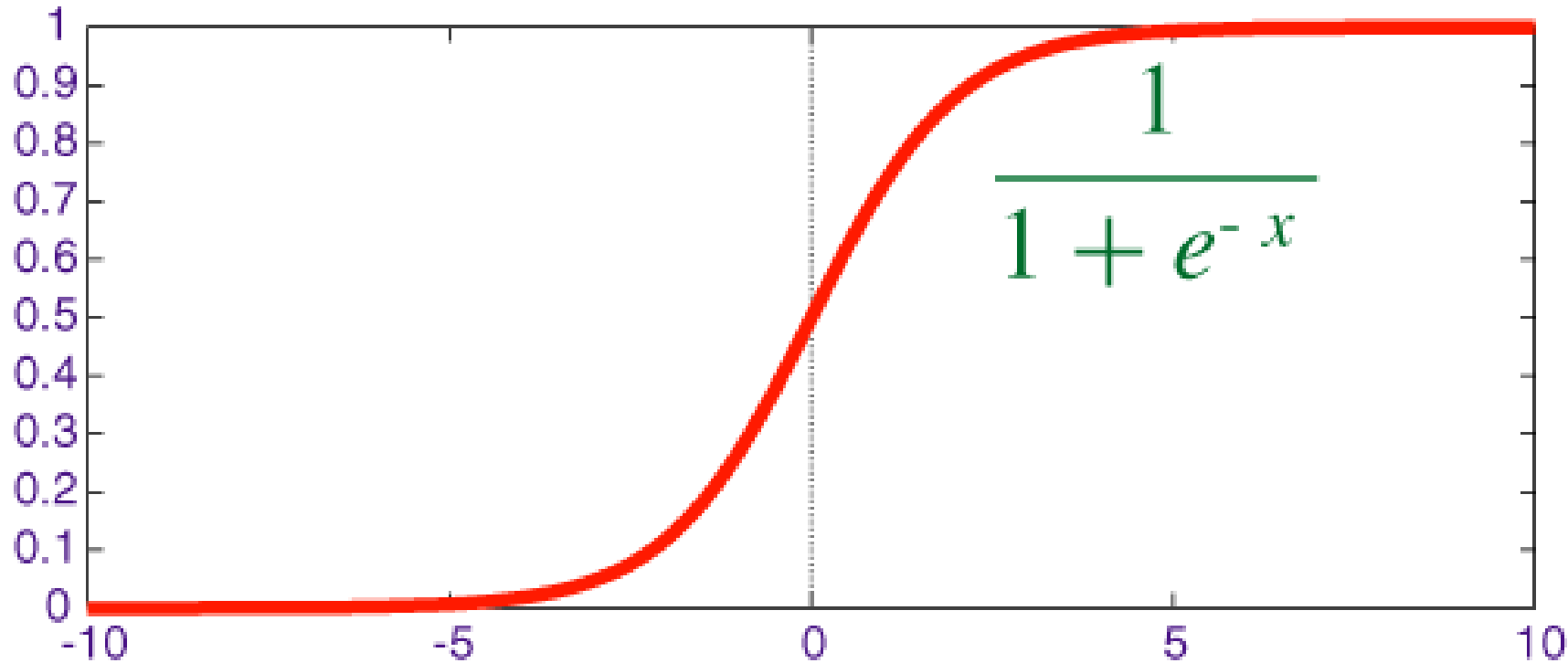
<https://www.javatpoint.com/machine-learning-polynomial-regression>

- Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance of belonging to a given class or not. It is a kind of statistical algorithm, which analyze the relationship between a set of independent variables and the dependent binary variables .For example, email spam or not.

<https://www.geeksforgeeks.org/understanding-logistic-regression/>

Logistic Function (Sigmoid Function)

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.

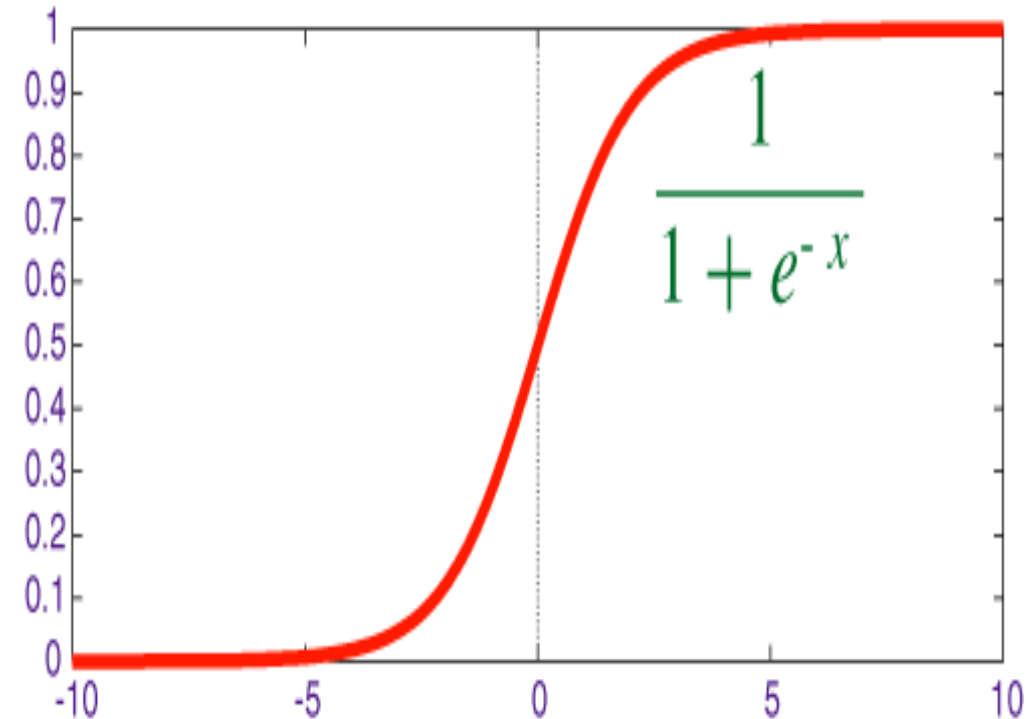


Logistic Function (Sigmoid Function)

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.

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

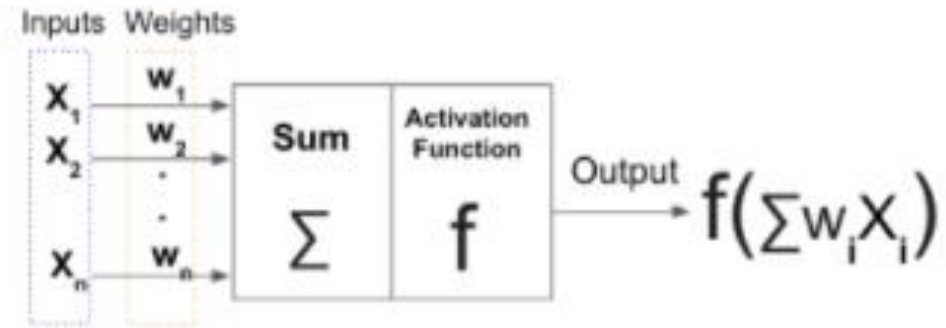
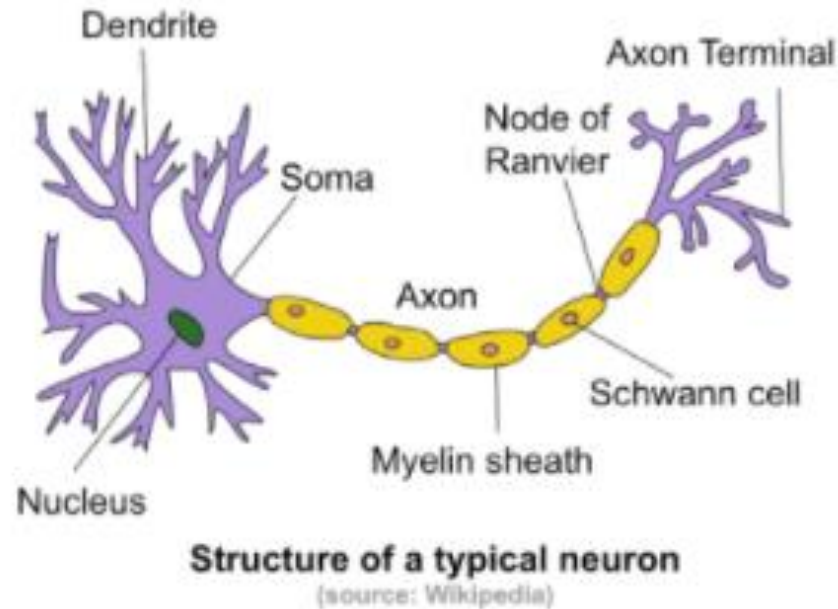
- ***e***:
 - [Number of Euler](#)
 - The base of the natural logarithms.
 - This constant is approximately 2.718
- ***x***: Can be the output of your **Neural Network**



- Frank Rosenblatt (1928 – 1971) was an American psychologist notable in the field of Artificial Intelligence.
- In 1957 he started something really big. He "invented" a Perceptron program, on an IBM 704 computer at Cornell Aeronautical Laboratory.

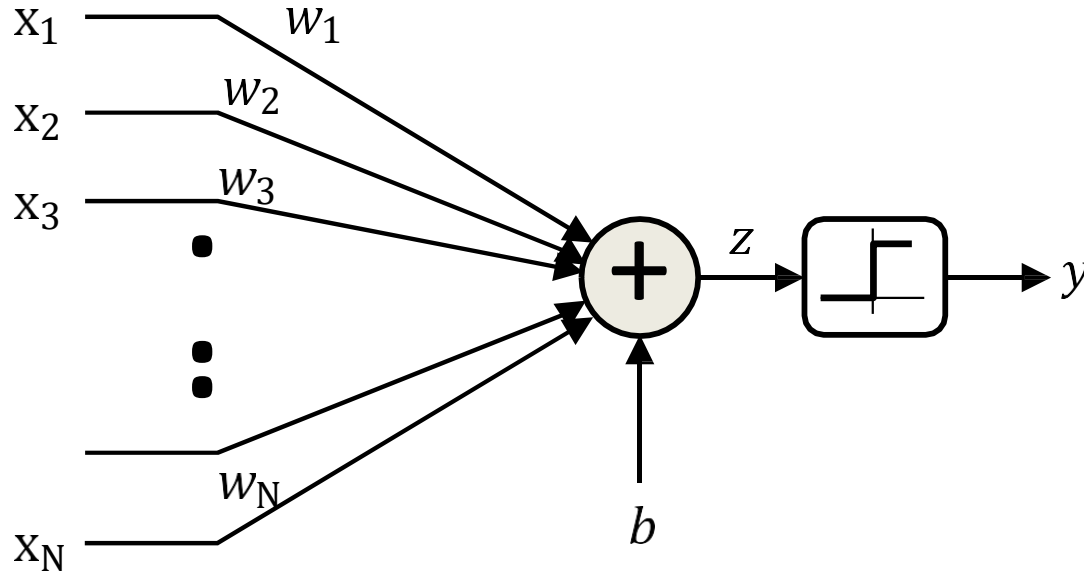


Biological Neurons



Structure of artificial neuron

Image credit: [Wikipedia](https://en.wikipedia.org/wiki/Neuron)



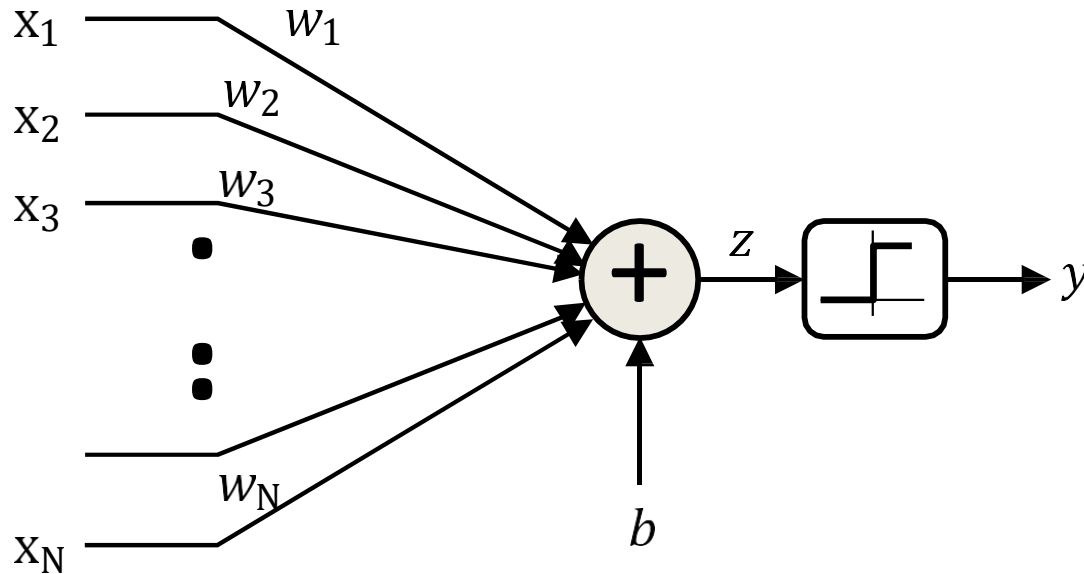
Linear vs Affine?

$$z = \sum_i w_i x_i + b$$

$$y = \begin{cases} 1 & \text{if } z \geq T \\ 0 & \text{else} \end{cases}$$

- A threshold unit
 - “Fires” if the affine function of inputs is positive
 - The bias is the negative of the threshold T
 - “Fires” if the weighted sum of inputs exceeds a threshold

<https://deeplearning.cs.cmu.edu/S23/>



Linear vs Affine?

$$z = \sum_i w_i x_i + b$$

$$y = \begin{cases} 1 & \text{if } z \geq T \\ 0 & \text{else} \end{cases}$$

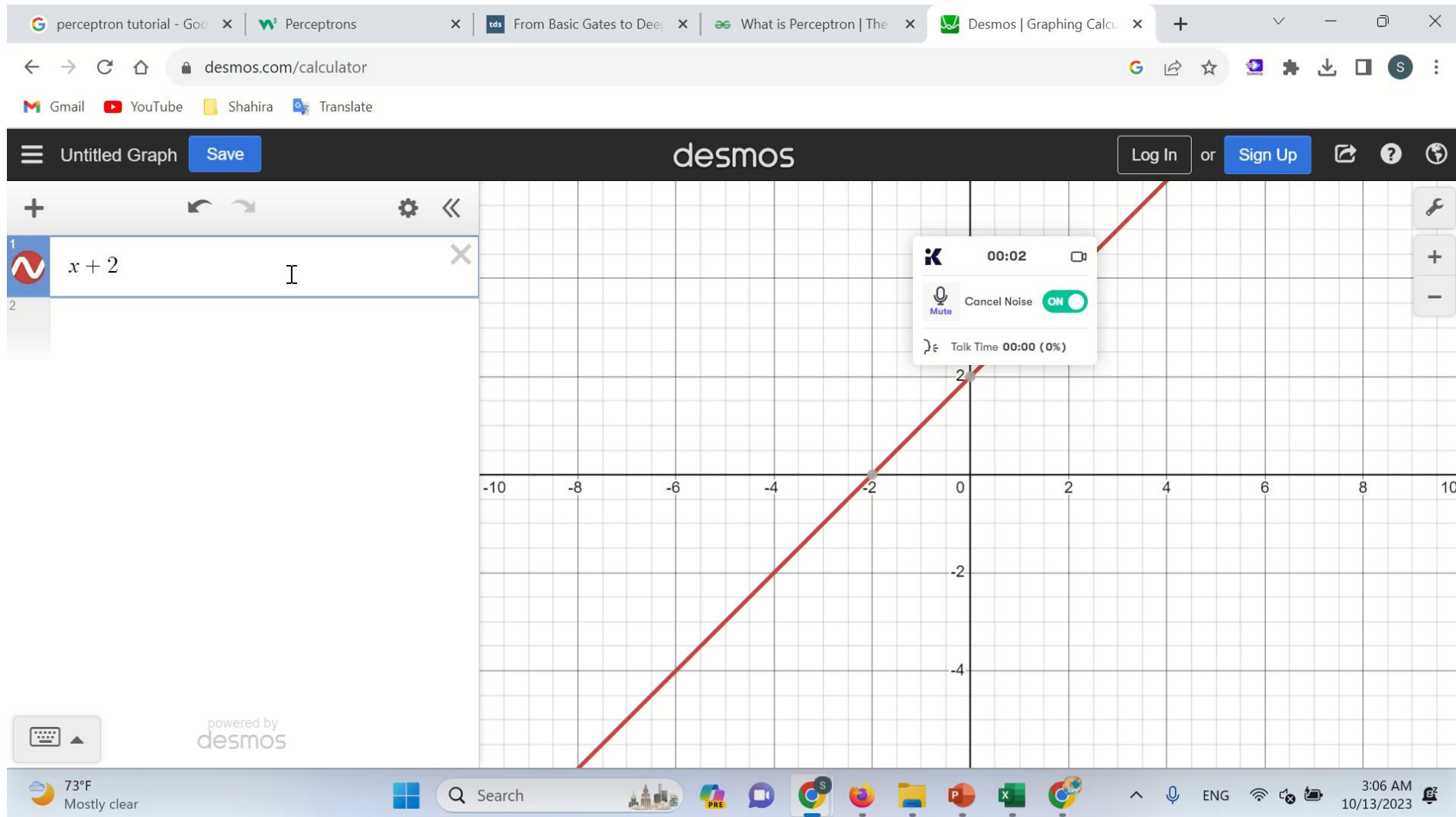
- A threshold unit
 - “Fires” if the affine function of inputs is positive
 - The bias is the negative of the threshold T
 - “Fires” if the weighted sum of inputs exceeds a threshold

<https://deeplearning.cs.cmu.edu/S23/>

- Mark all true statements
- $3x + 7y$ is a linear combination of x and y
- $3x + 7y + 4$ is a linear combination of x and y
- $3x + 7y$ is an affine function of x and y
- $3x + 7y + 4$ is an affine function of x and y

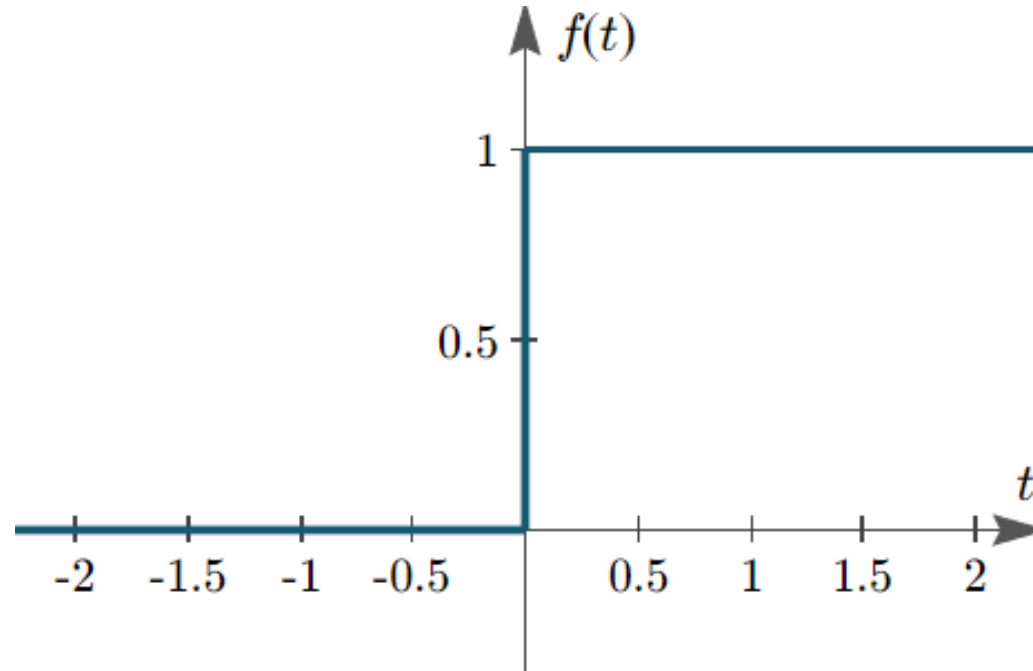
- Mark all true statements
 - $3x + 7y$ is a linear combination of x and y
 - $3x + 7y + 4$ is a linear combination of x and y
 - $3x + 7y$ is an affine function of x and y
 - $3x + 7y + 4$ is an affine function of x and y

Linear vs Affine

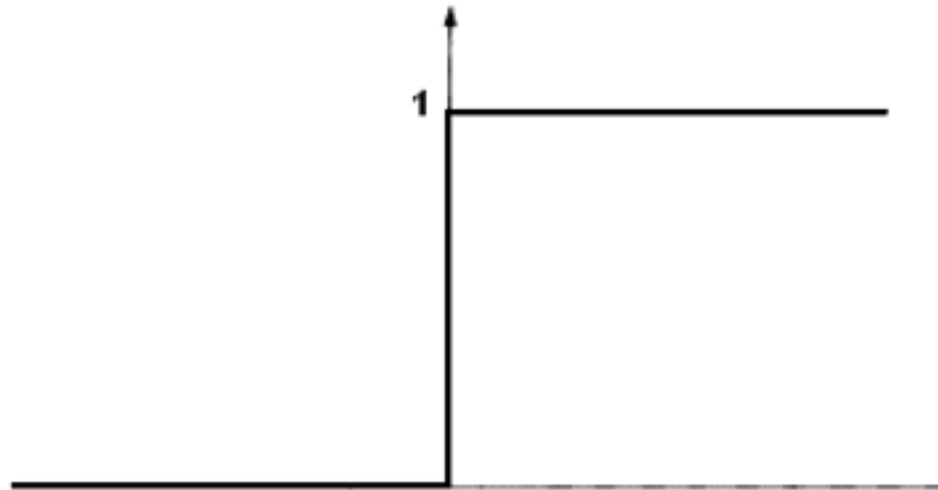


The Unit Step Function (Heaviside Function)

$$u(t) = \begin{cases} 0 & t < 0 \\ 1 & t > 0 \end{cases}$$



Threshold Function



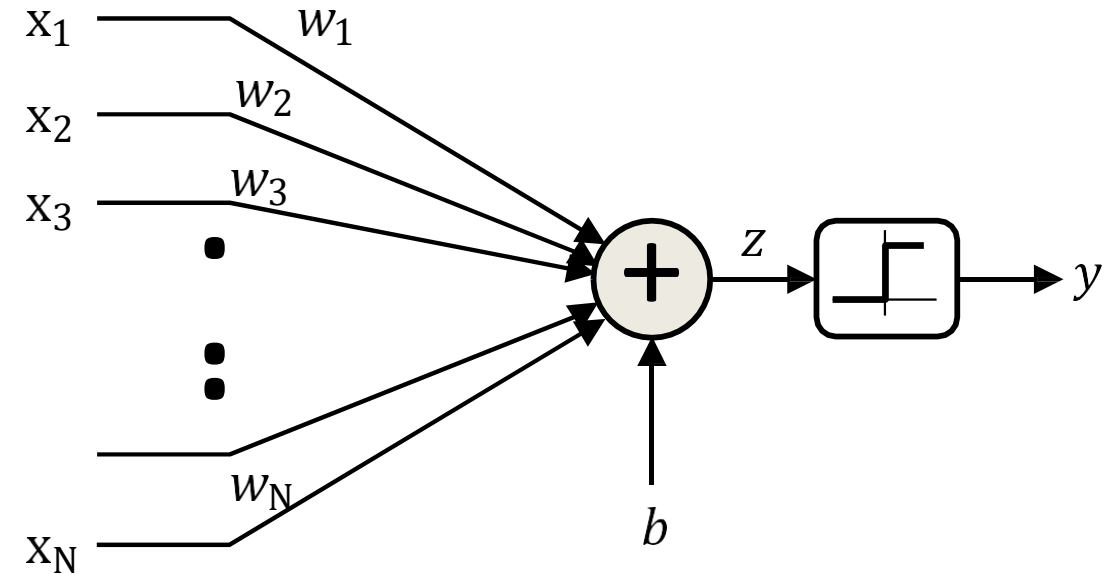
$$y = \begin{cases} 1 & \text{if } z \geq \tau \\ 0 & \text{else} \end{cases}$$

- **Set a threshold value:**
 - Threshold = 1.5

Input	Weight
$x1 = 0 \text{ or } 1$	$w1 = 0.7$
$x2 = 0 \text{ or } 1$	$w2 = 0.6$
$x3 = 0 \text{ or } 1$	$w3 = 0.5$
$x4 = 0 \text{ or } 1$	$w4 = 0.3$
$x5 = 0 \text{ or } 1$	$w5 = 0.4$

Perceptron Example

- **Set a threshold value:**
 - Threshold = 1.5
 - Input = [10101]
- **Multiply all inputs with its weights:**
 - $x_1 * w_1 = 1 * 0.7 = 0.7$
 - $x_2 * w_2 = 0 * 0.6 = 0$
 - $x_3 * w_3 = 1 * 0.5 = 0.5$
 - $x_4 * w_4 = 0 * 0.3 = 0$
 - $x_5 * w_5 = 1 * 0.4 = 0.4$



Input	Weight
$x_1 = 0 \text{ or } 1$	$w_1 = 0.7$
$x_2 = 0 \text{ or } 1$	$w_2 = 0.6$
$x_3 = 0 \text{ or } 1$	$w_3 = 0.5$
$x_4 = 0 \text{ or } 1$	$w_4 = 0.3$
$x_5 = 0 \text{ or } 1$	$w_5 = 0.4$

Perceptron Example

- **Set a threshold value:**

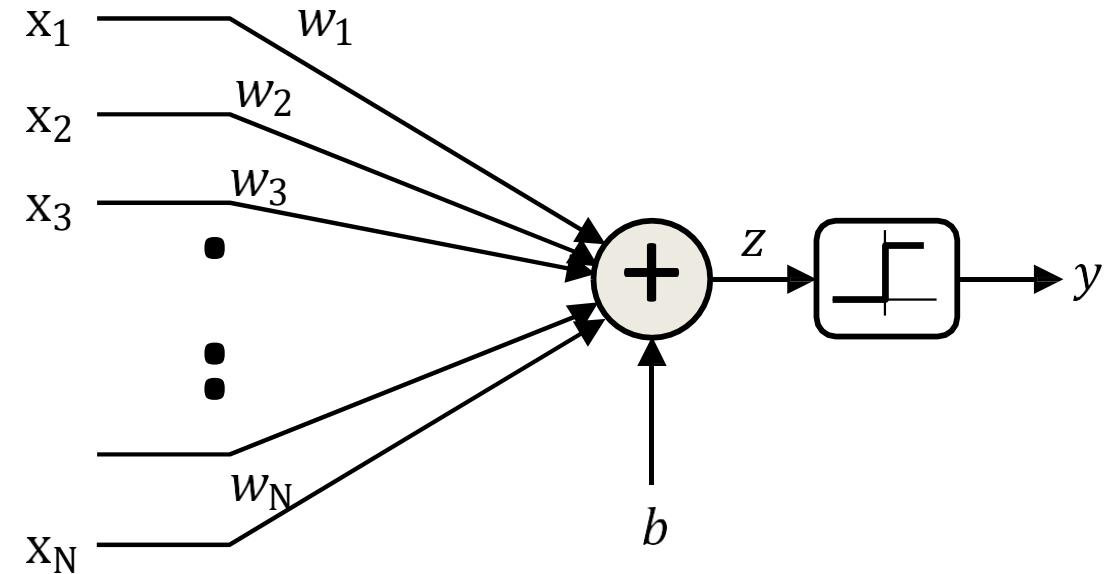
- Threshold = 1.5
- Input = [10101]

- **Multiply all inputs with its weights:**

- $x_1 * w_1 = 1 * 0.7 = 0.7$
- $x_2 * w_2 = 0 * 0.6 = 0$
- $x_3 * w_3 = 1 * 0.5 = 0.5$
- $x_4 * w_4 = 0 * 0.3 = 0$
- $x_5 * w_5 = 1 * 0.4 = 0.4$

- **Sum all the results:**

- $0.7 + 0 + 0.5 + 0 + 0.4 = 1.6$



Input	Weight
$x_1 = 0 \text{ or } 1$	$w_1 = 0.7$
$x_2 = 0 \text{ or } 1$	$w_2 = 0.6$
$x_3 = 0 \text{ or } 1$	$w_3 = 0.5$
$x_4 = 0 \text{ or } 1$	$w_4 = 0.3$
$x_5 = 0 \text{ or } 1$	$w_5 = 0.4$

Perceptron Example

- **Set a threshold value:**

- Threshold = 1.5
- Input = [10101]

- **Multiply all inputs with its weights:**

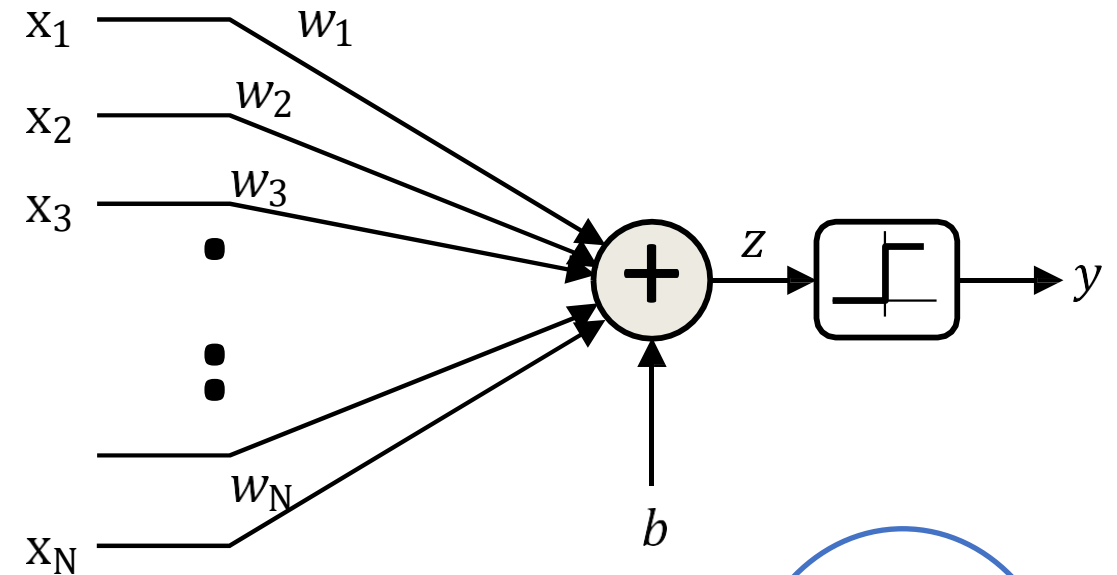
- $x_1 * w_1 = 1 * 0.7 = 0.7$
- $x_2 * w_2 = 0 * 0.6 = 0$
- $x_3 * w_3 = 1 * 0.5 = 0.5$
- $x_4 * w_4 = 0 * 0.3 = 0$
- $x_5 * w_5 = 1 * 0.4 = 0.4$

- **Sum all the results:**

- $0.7 + 0 + 0.5 + 0 + 0.4 = 1.6$

- **Activate the Output:**

- Return true if the sum > 1.5



Input	Weight
$x_1 = 0 \text{ or } 1$	$w_1 = 0.7$
$x_2 = 0 \text{ or } 1$	$w_2 = 0.6$
$x_3 = 0 \text{ or } 1$	$w_3 = 0.5$
$x_4 = 0 \text{ or } 1$	$w_4 = 0.3$
$x_5 = 0 \text{ or } 1$	$w_5 = 0.4$

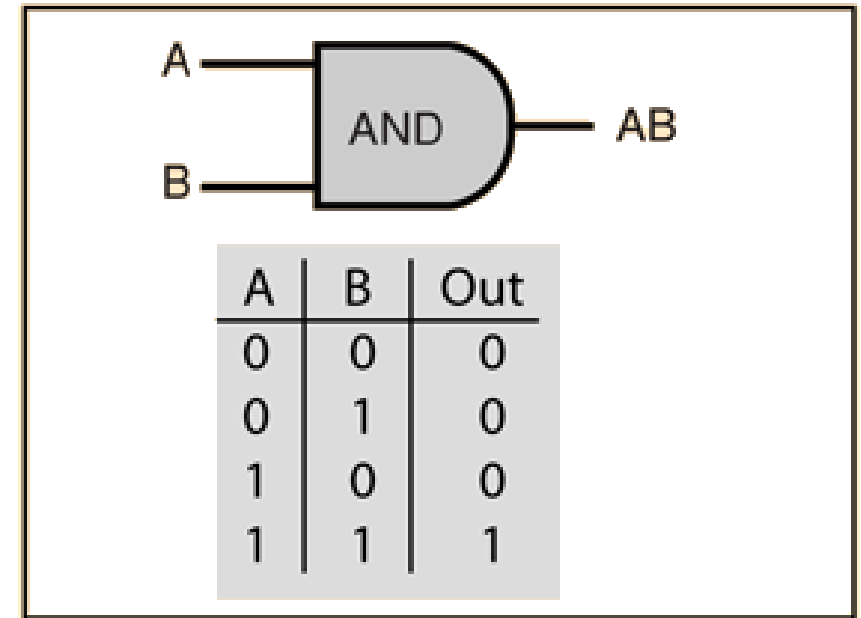
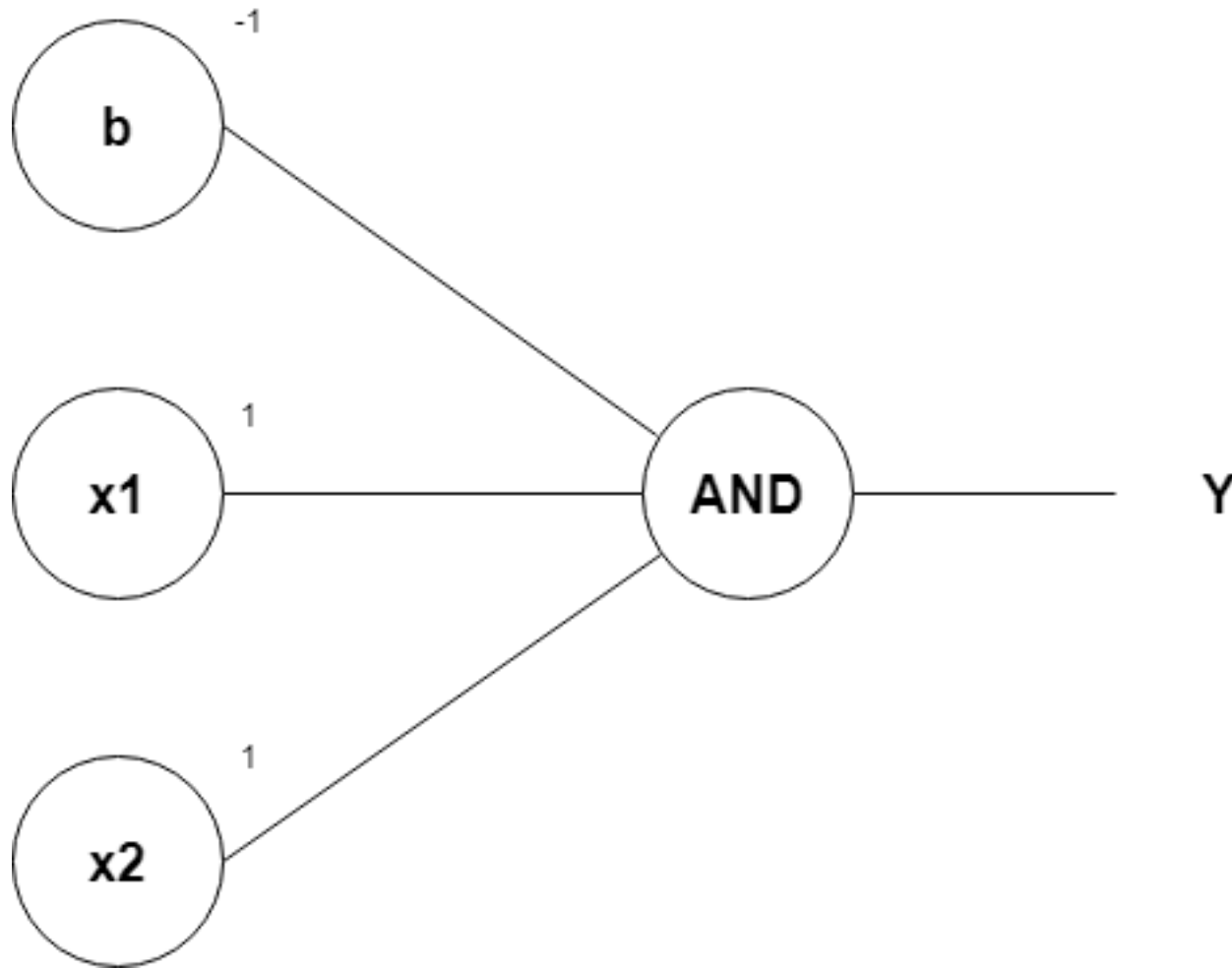




الجامعة
البحرية
ميدوcean

Neural Representation of AND, OR, NOT, XOR and XNOR Logic Gates

AND Gate

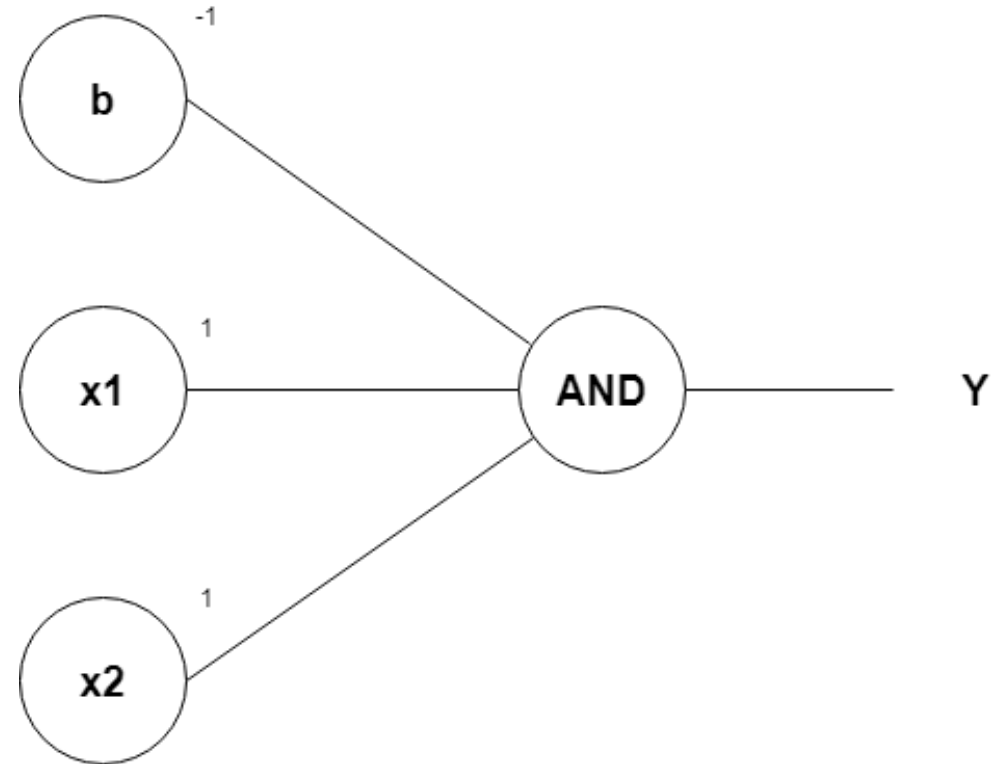


- Row 1

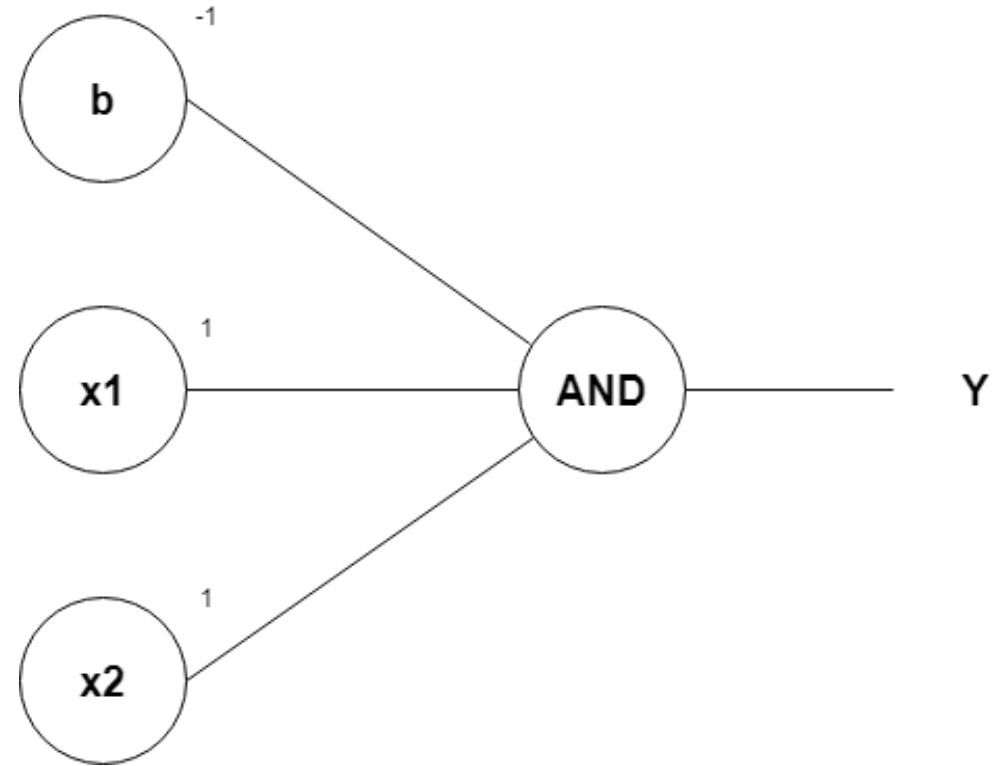
From $w_1 \cdot x_1 + w_2 \cdot x_2 + b$, initializing w_1 , w_2 , as 1 and b as -1 , we get;

$$x_1(1) + x_2(1) - 1$$

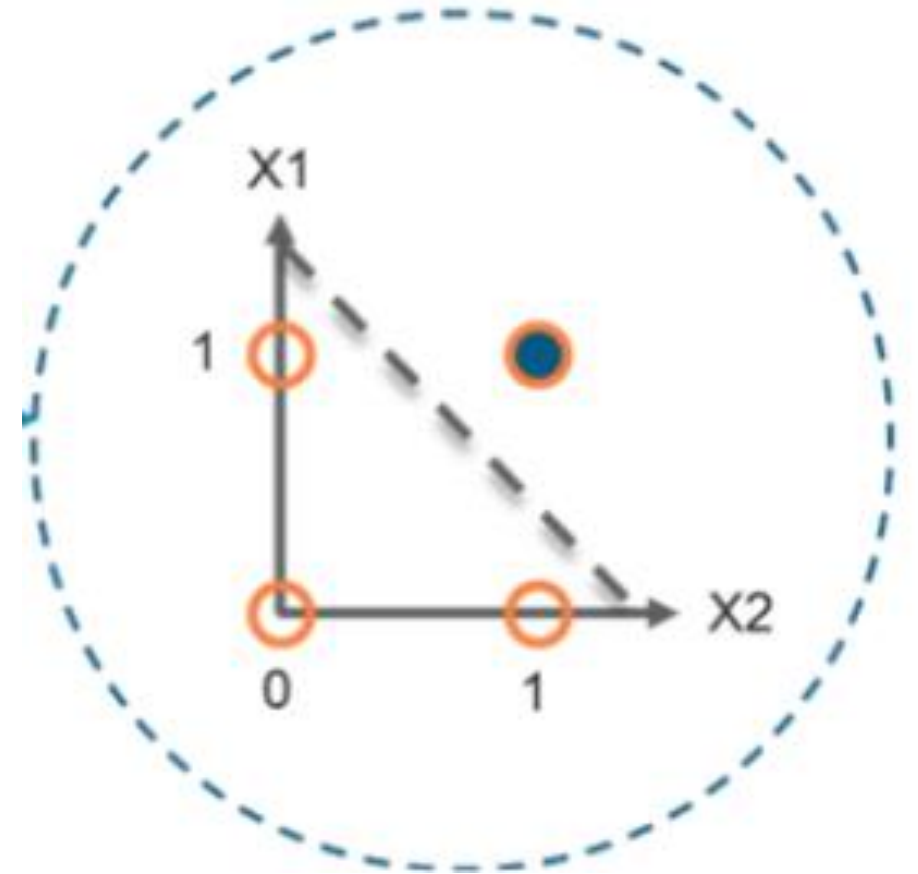
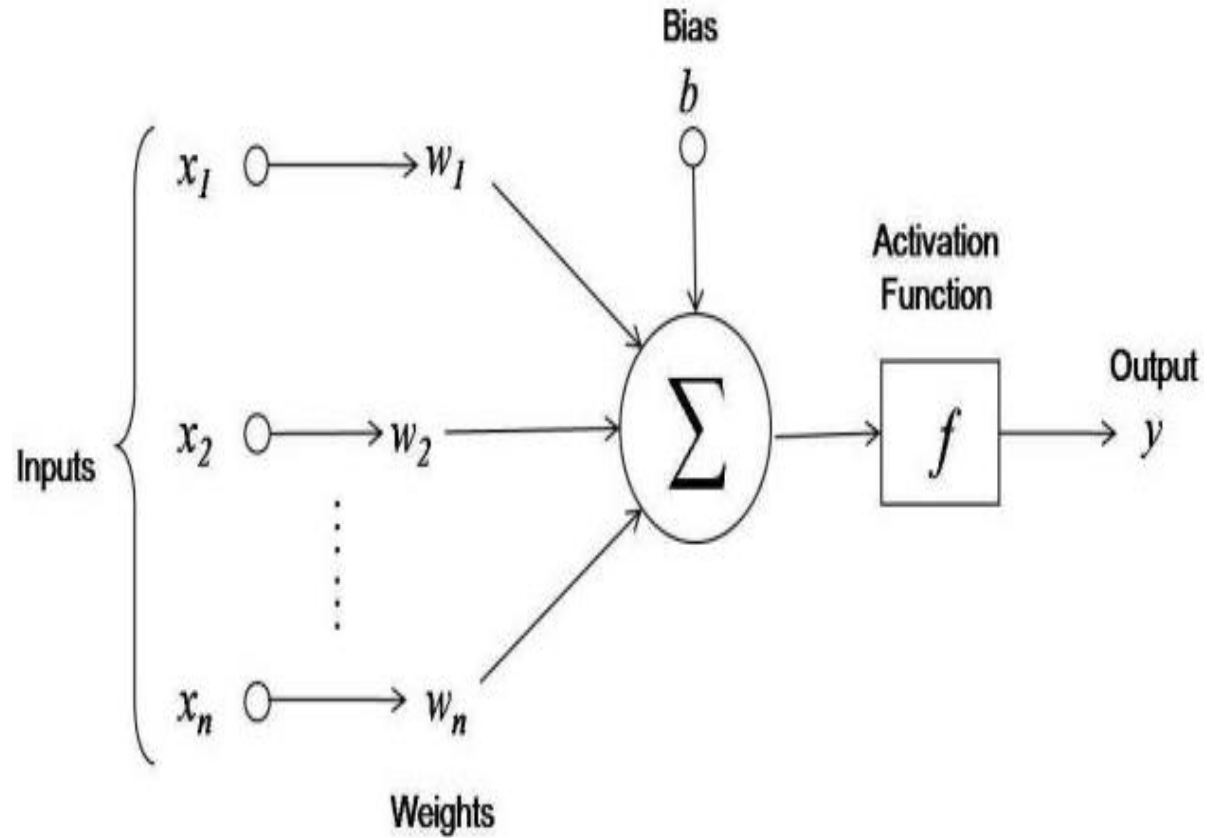
- Passing the first row of the AND logic table ($x_1=0$, $x_2=0$), we get;
- $0+0-1 = -1$
- From the Perceptron rule, if $Wx+b \leq 0$, then $y'=0$.



- Passing ($x_1=0$ and $x_2=1$), we get;
 $0+1-1 = 0$
 - From the Perceptron rule, if $Wx+b \leq 0$, then $y'=0$.
- Passing ($x_1=1$ and $x_2=0$), we get;
 $1+0-1 = 0$
- Passing ($x_1=1$ and $x_2=1$), we get;
 $1+1-1 = 1$



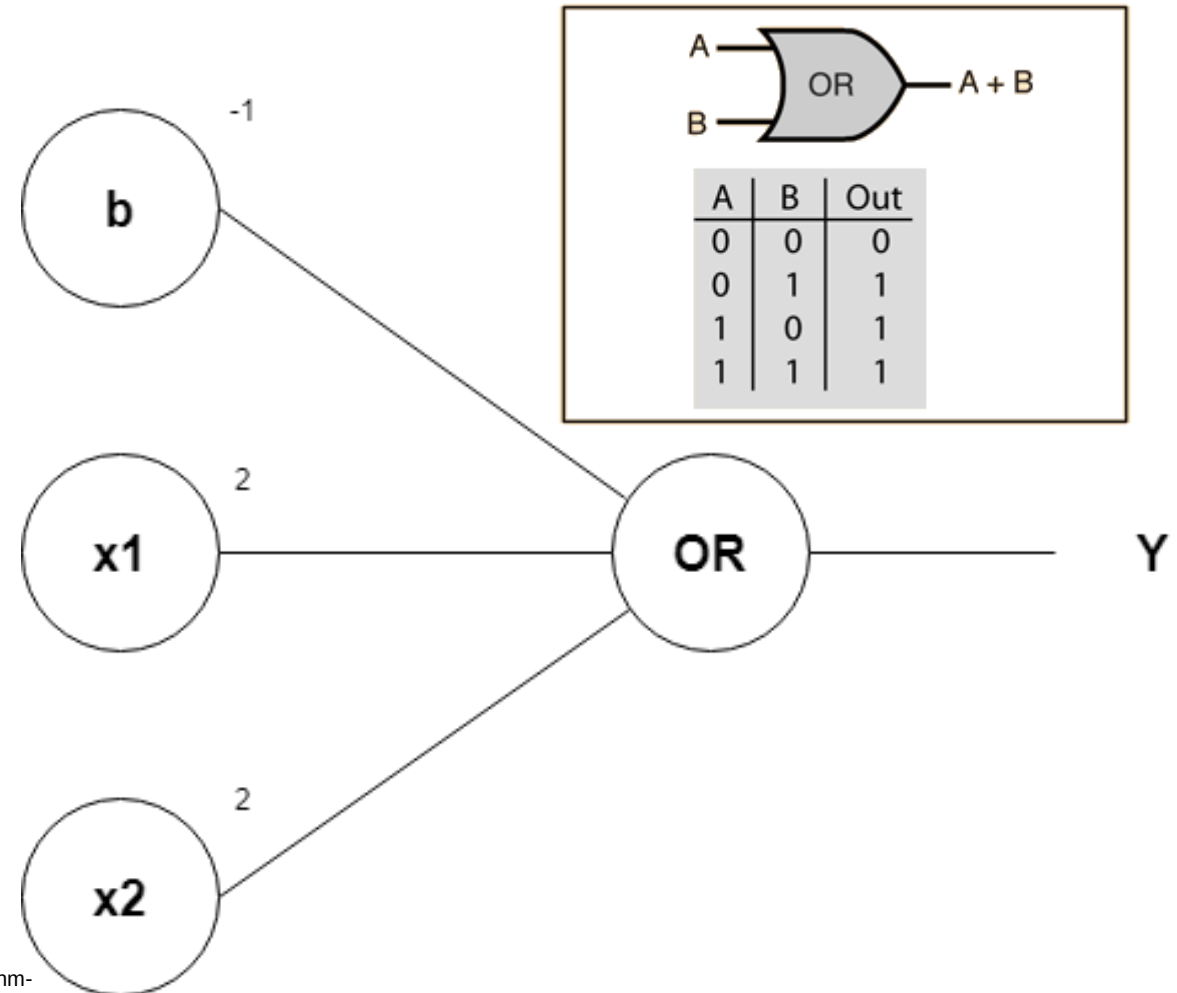
AND Gate



<https://medium.com/@stanleydukor/neural-representation-of-and-or-not-xor-and-xnor-logic-gates-perceptron-algorithm-b0275375fea1>

OR Gate

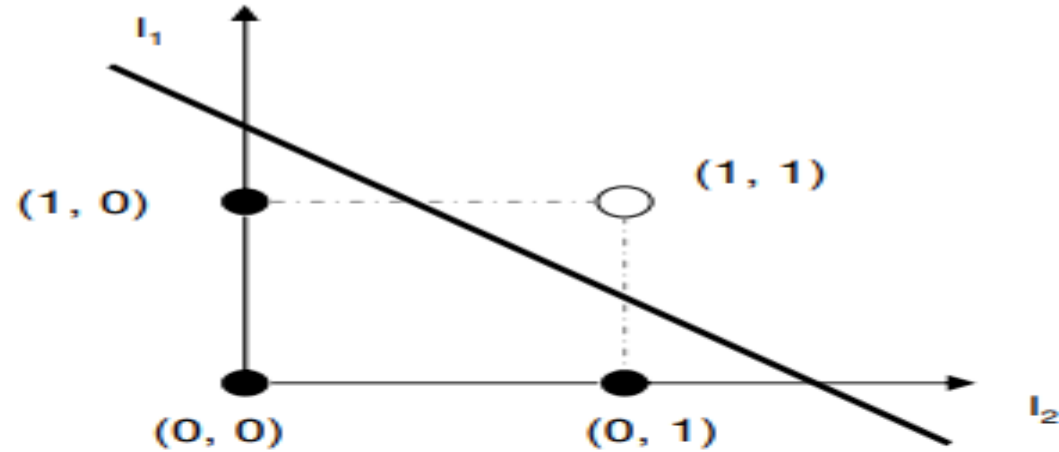
- From $w_1x_1 + w_2x_2 + b$, initializing w_1 , w_2 , as 1 and b as -1, we get $x_1(2) + x_2(2) - 1$
- $0, 0 \rightarrow x_1(2) + x_2(2) - 1 \rightarrow 0(2) + 0(2) - 1 = -1$
- $0, 1 \rightarrow x_1(2) + x_2(2) - 1 \rightarrow 0(2) + 1(2) - 1 = 1$
- $1, 0 \rightarrow x_1(2) + x_2(2) - 1 \rightarrow 1(2) + 0(2) - 1 = 1$
- $1, 1 \rightarrow x_1(2) + x_2(2) - 1 \rightarrow 1(2) + 1(2) - 1 = 3$



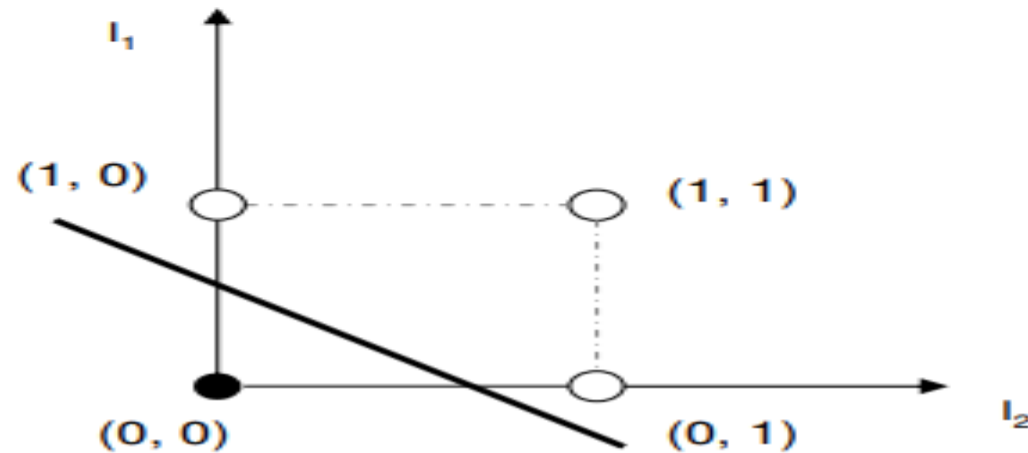
<https://medium.com/@stanleydukor/neural-representation-of-and-or-not-xor-and-xnor-logic-gates-perceptron-algorithm-b0275375fea1>

AND Gate

AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1

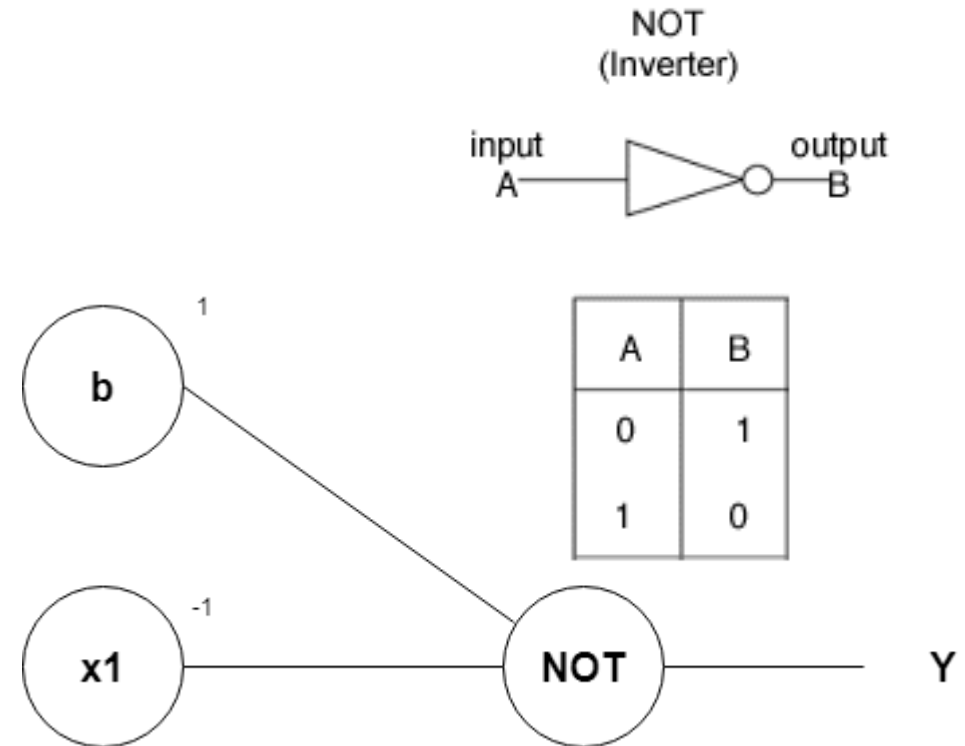


OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1



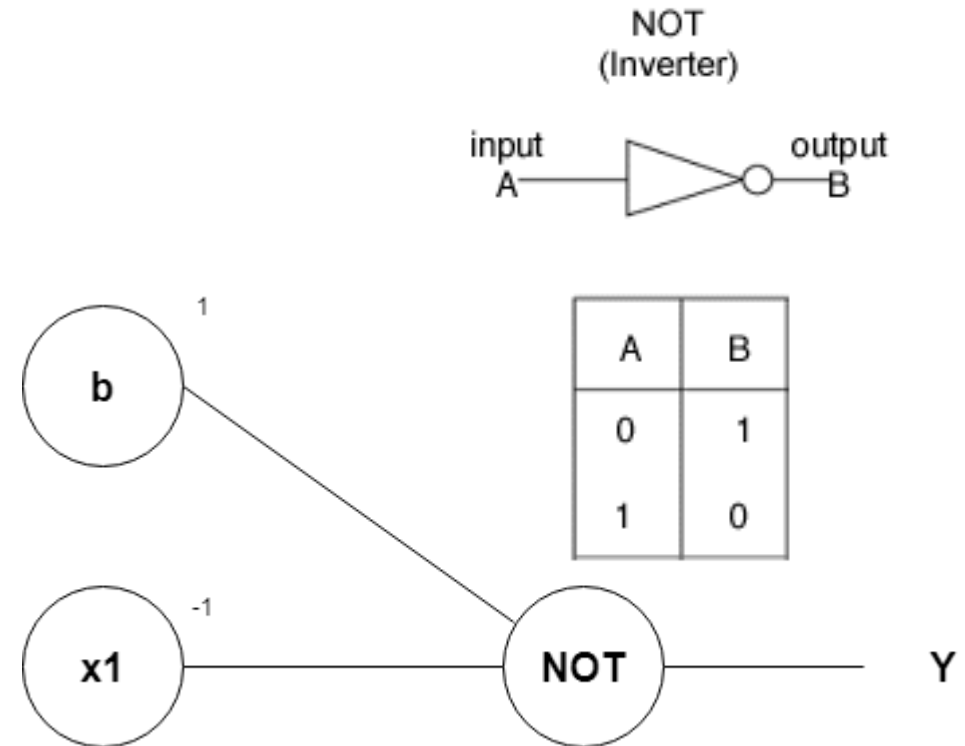
Not Gate

- From w_1x_1+b , initializing w_1 as 1 (since single input), and b as -1 , we get; $x_1(-1)+1$
- Passing ($x_1=0$), we get;
- $0-1 = -1$
- Passing ($x_1=1$), we get;
- $-1+1 = 0$

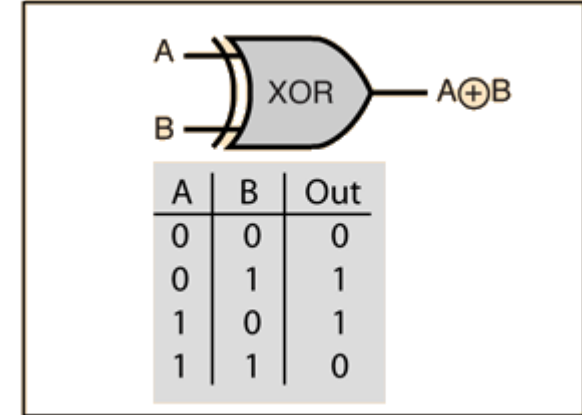


Not Gate

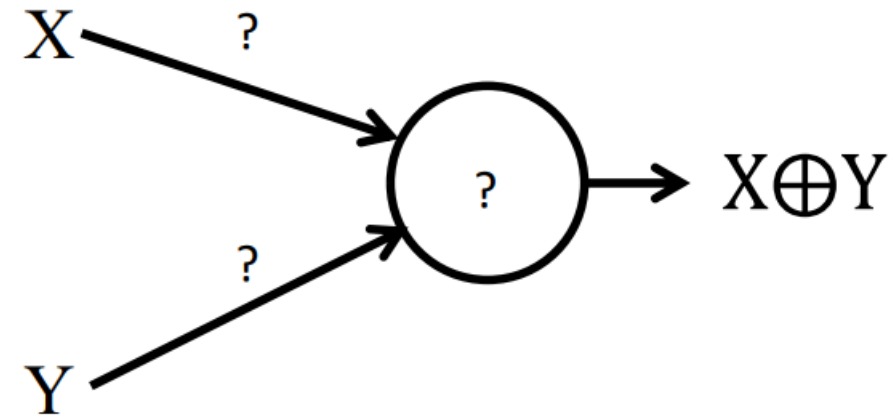
- From w_1x_1+b , initializing w_1 as 1 (since single input), and b as -1 , we get; $x_1(-1)+1$
- Passing ($x_1=0$), we get;
- $0-1 = -1$
- Passing ($x_1=1$), we get;
- $-1+1 = 0$



- Cannot be represented with a single perceptron

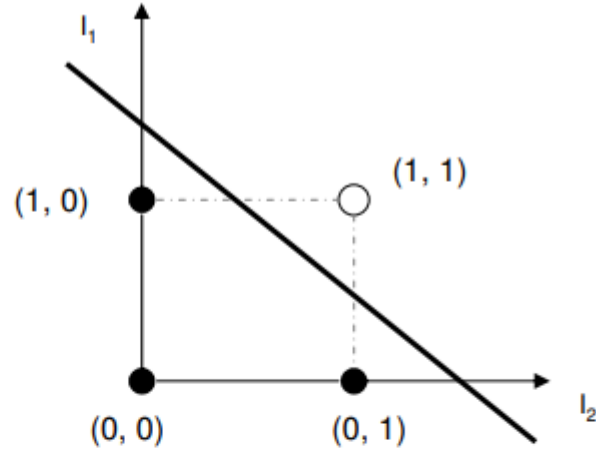


- The perceptron is not enough

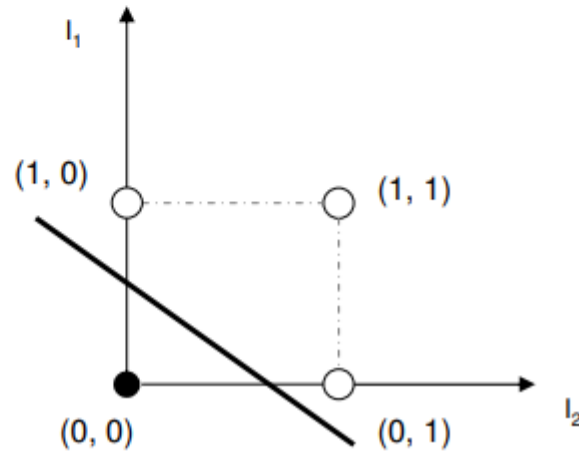


XOR Gate

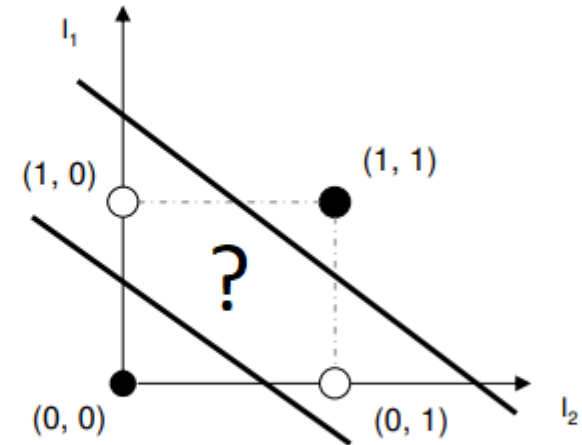
AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



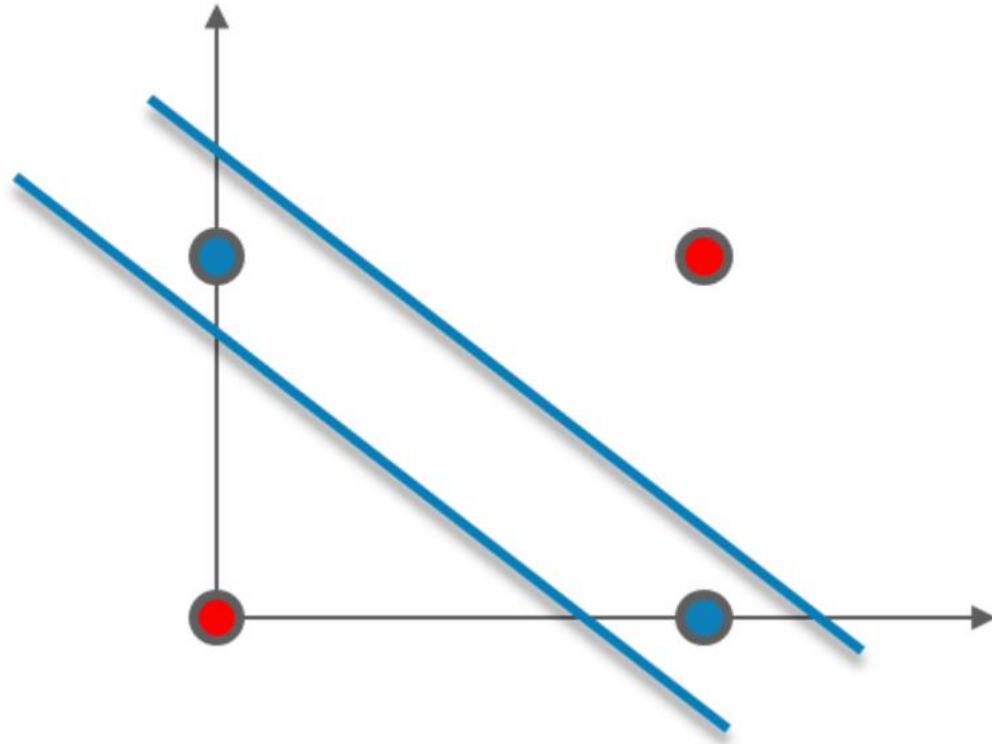
OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1



XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0



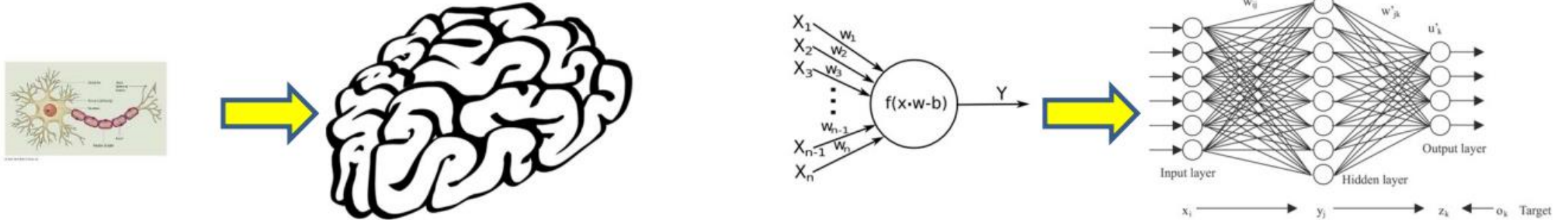
- Cannot be represented with a single perceptron



- Frank Rosenblatt (1928 – 1971) was an American psychologist notable in the field of Artificial Intelligence.
- In 1957 he started something really big. He "invented" a Perceptron program, on an IBM 704 computer at Cornell Aeronautical Laboratory.

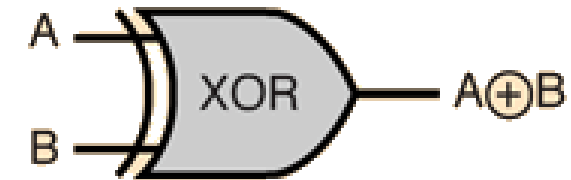
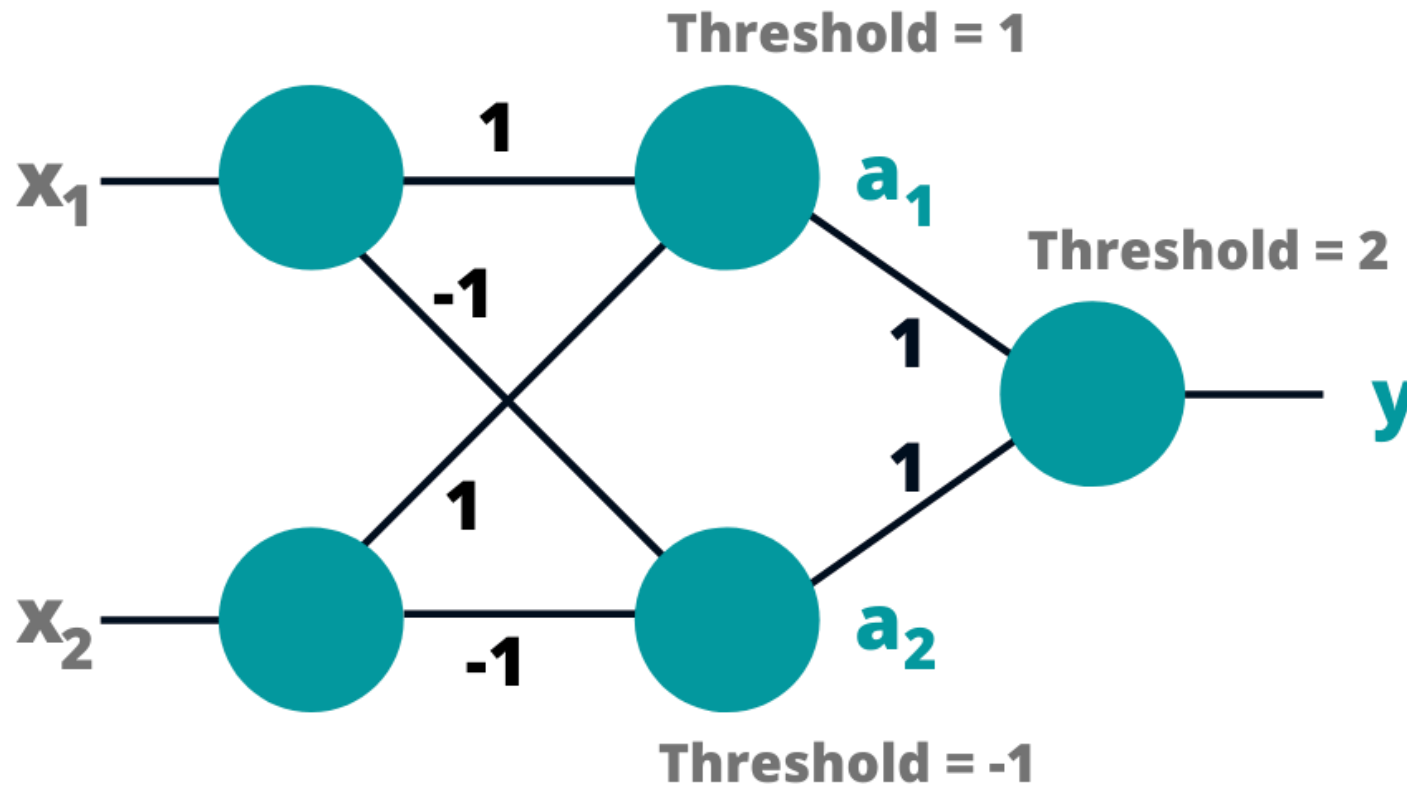


A single Neuron is not enough



<https://deeplearning.cs.cmu.edu/S23/document/slides/lec1.intro.pdf>

Multilayer Perceptron



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

<https://www.deeplearningnerds.com/Bilder/xor-problem-neural-network.png>

Multilayer Perceptron-XOR Gate

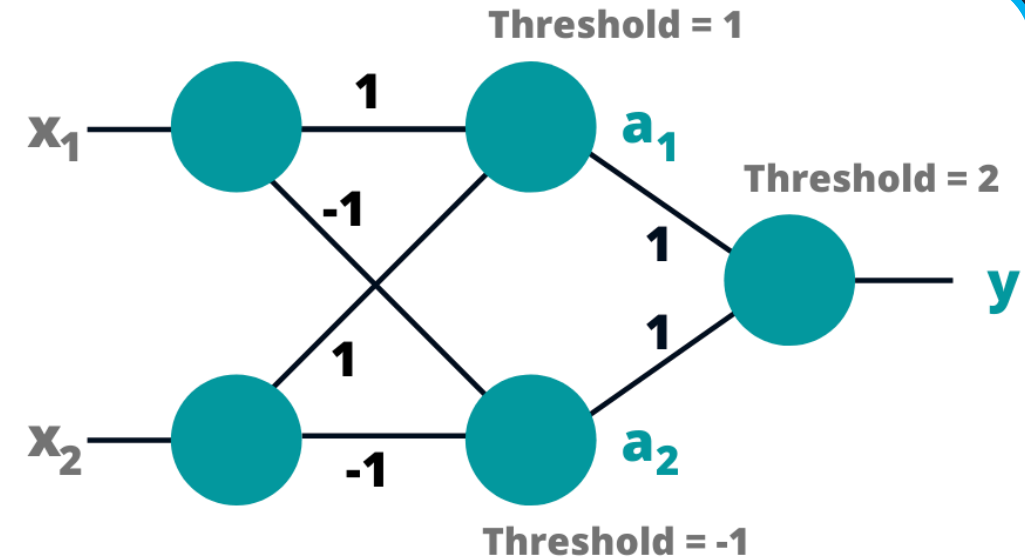
Case 1

the inputs $x_1 = 0$ and $x_2 = 0$ and the output should be $y = 0$

Hidden Unit 1: $x_1 + x_2 = 0 + 0 = 0 \Rightarrow a_1 = 0$

Hidden Unit 2: $-x_1 - x_2 = 0 - 0 = 0 \Rightarrow a_2 = 1$

Output Unit: $a_1 + a_2 = 0 + 1 = 1 \Rightarrow y = 0$



<https://www.deeplearningnerds.com/Bilder/xor-problem-neural-network.png>

Multilayer Perceptron-XOR Gate

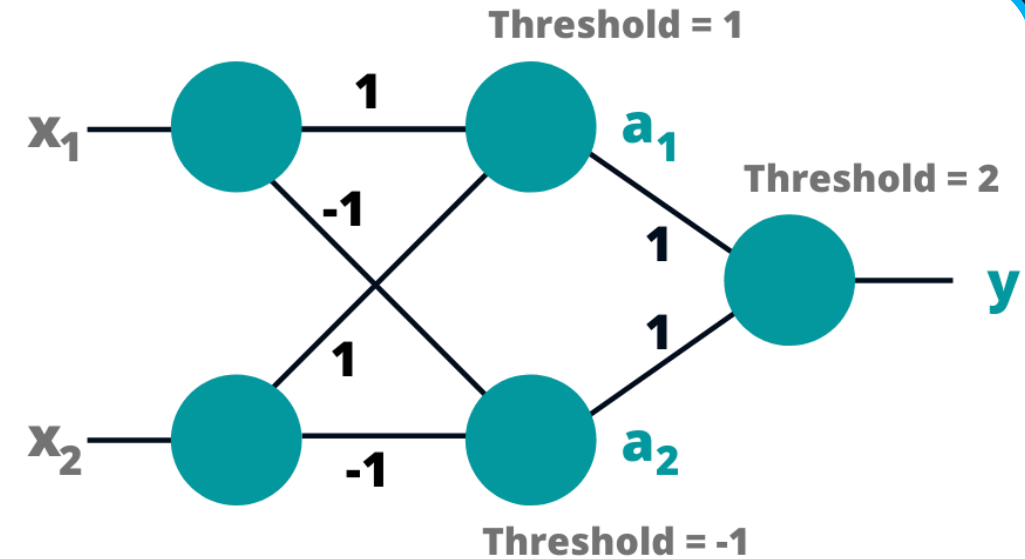
Case 2

the inputs $x_1 = 0$ and $x_2 = 1$ and the output should be $y = 1$.

Hidden Unit 1: $x_1 + x_2 = 0 + 1 = 1 \Rightarrow a_1 = 1$

Hidden Unit 2: $-x_1 - x_2 = 0 - 1 = -1 \Rightarrow a_2 = 1$

Output Unit: $a_1 + a_2 = 1 + 1 = 2 \Rightarrow y = 1$



Multilayer Perceptron-XOR Gate

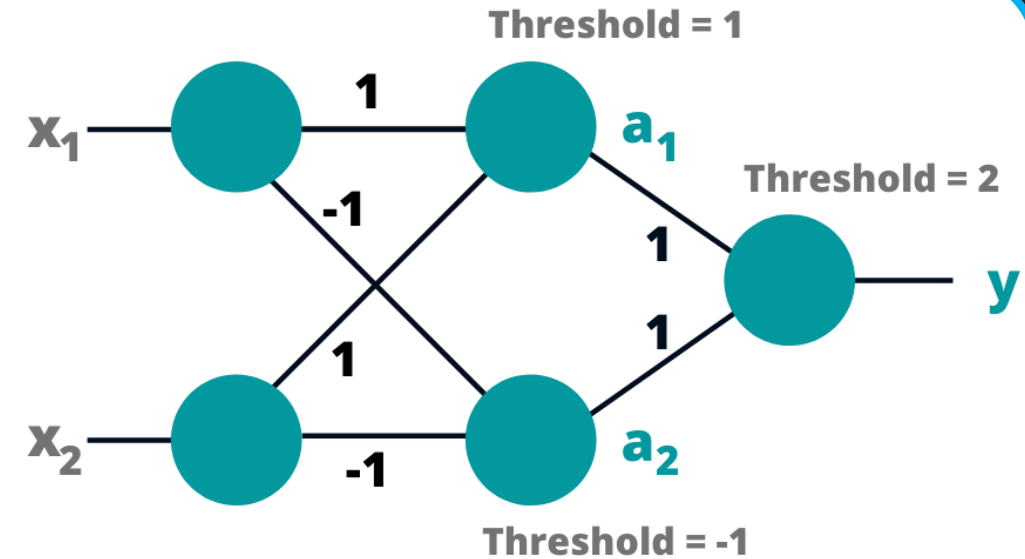
Case 3

the inputs $x_1 = 1$ and $x_2 = 0$ and the output should be $y = 1$.

Hidden Unit 1: $x_1 + x_2 = 1 + 0 = 1 \Rightarrow a_1 = 1$

Hidden Unit 2: $-x_1 - x_2 = -1 - 0 = -1 \Rightarrow a_2 = 1$

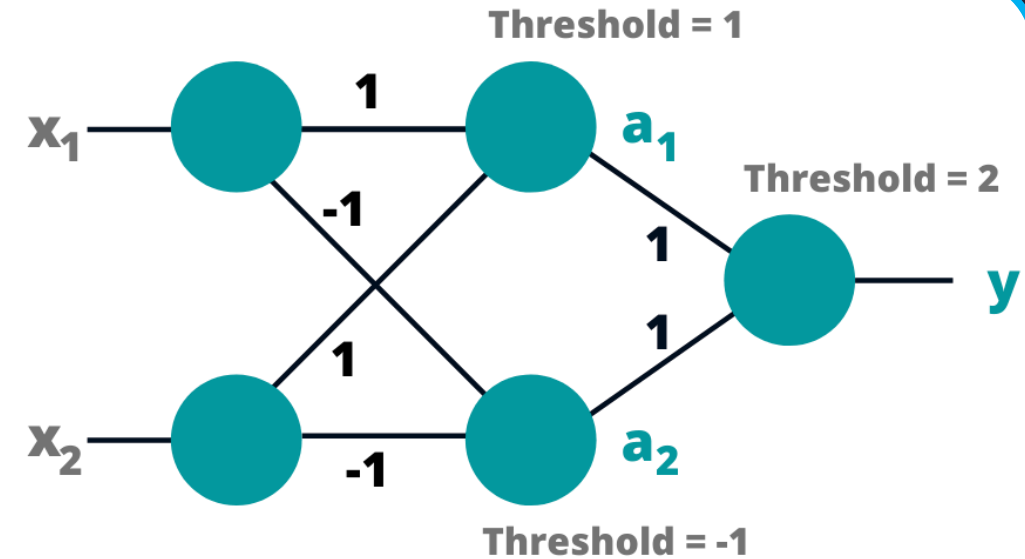
Output Unit: $a_1 + a_2 = 1 + 1 = 2 \Rightarrow y = 1$



Multilayer Perceptron-XOR Gate

Case 4

the inputs $x_1 = 1$ and $x_2 = 1$ and the output should be $y = 0$.



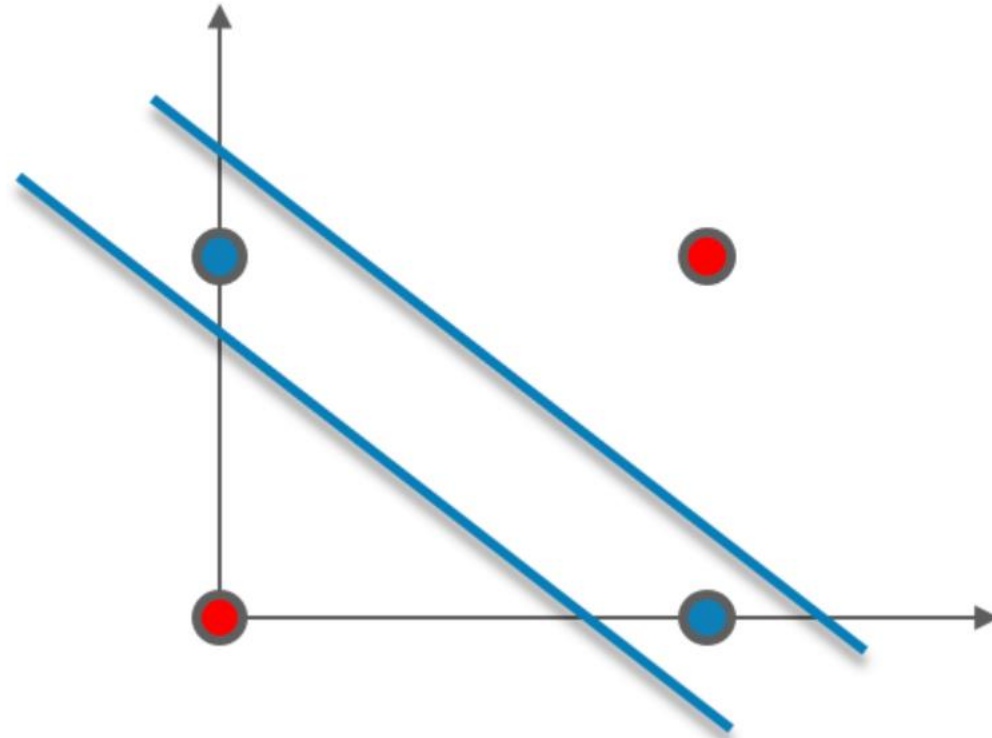
Hidden Unit 1: $x_1 + x_2 = 1 + 1 = 2 \Rightarrow a_1 = 1$

Hidden Unit 2: $-x_1 - x_2 = -1 - 1 = -2 \Rightarrow a_2 = 0$

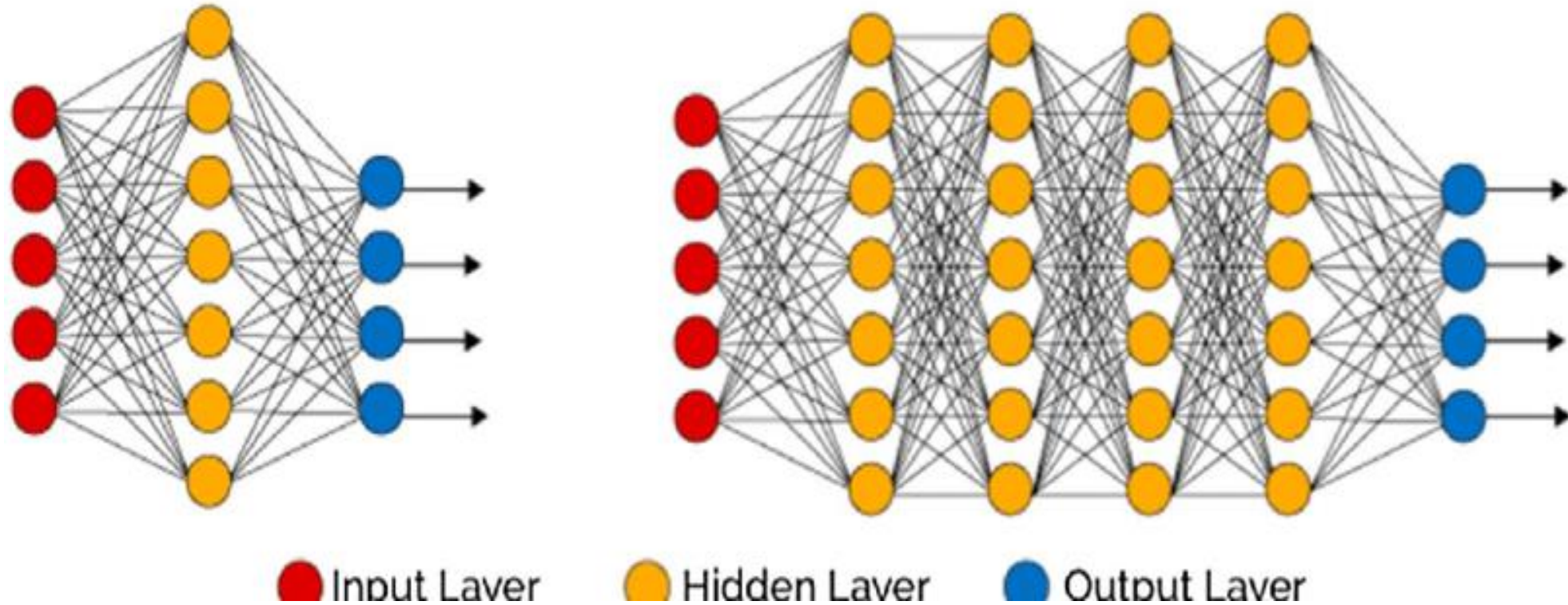
Output Unit: $a_1 + a_2 = 1 + 0 = 1 \Rightarrow y = 0$

<https://www.deeplearningnerds.com/Bilder/xor-problem-neural-network.png>

- XOR function can't be solved by a simple Perceptron. We need a Neural Network with as least one hidden layer.



Shallow learning vs Deep learning in neural network.



deep

Shallow

https://www.researchgate.net/figure/Shallow-MLP-vs-deep-MLP-57_fig1_332165523

Shallow learning vs Deep learning in neural network.

- The main difference between deep and shallow neural networks lies in the number of layers they contain.
- Shallow neural networks, also known as single-layer neural networks, consist of only one hidden layer between the input and output layers.
- On the other hand, deep neural networks have multiple hidden layers stacked between the input and output layers. These networks can be quite deep, with tens or even hundreds of layers. Each layer performs computations and passes the transformed data to the next layer. Deep neural networks have a greater capacity to learn more complex patterns and relationships.

<https://www.linkedin.com/pulse/what-difference-between-deep-shallow-neural-networks-anurodh-kumar/>

- Linear Regression
- Logistic Regression
- Perceptron
- Perceptron as logic gate
- Artificial Neural Network
 - Shallow
 - Deep



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY

*Thank
you*



www.midocean.ae

