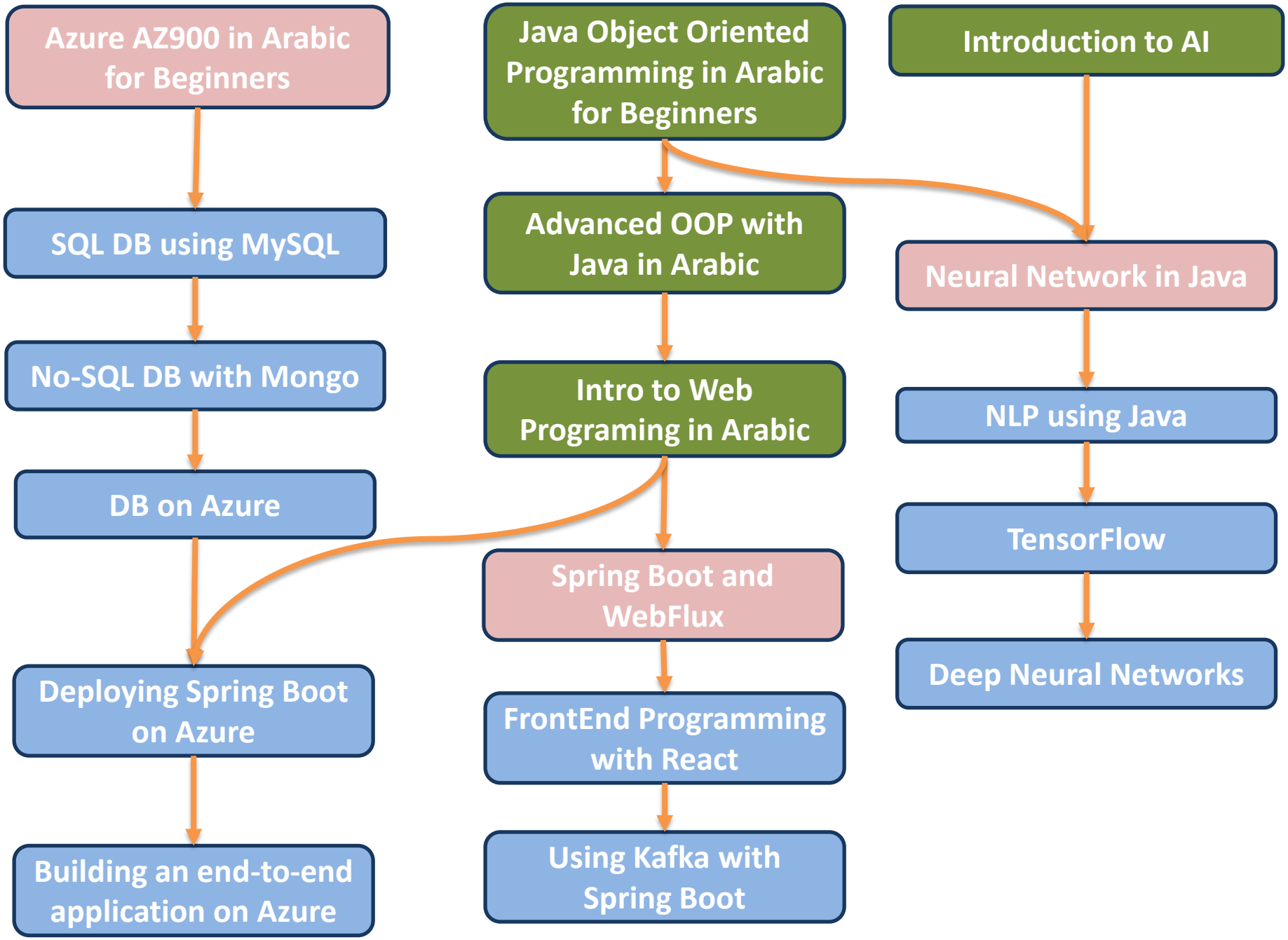
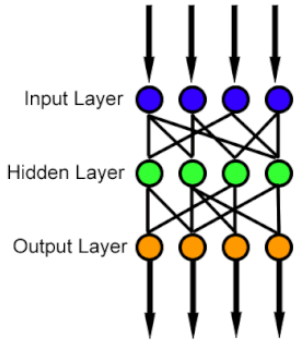

Azure AZ900 in Arabic for Beginners
SQL DB using MySQL
No-SQL DB with Mongo
DB on Azure
Deploying Spring Boot on Azure
Building an end-to-end application on Azure
Java Object Oriented Programming in Arabic for Beginners
Advanced OOP with Java in Arabic
Intro to Web Programing in Arabic
Spring Boot and WebFlux
FrontEnd Programming with React
Using Kafka with Spring Boot
Introduction to AI
Neural Network in Java
NLP using Java
TensorFlow
Deep Neural Networks

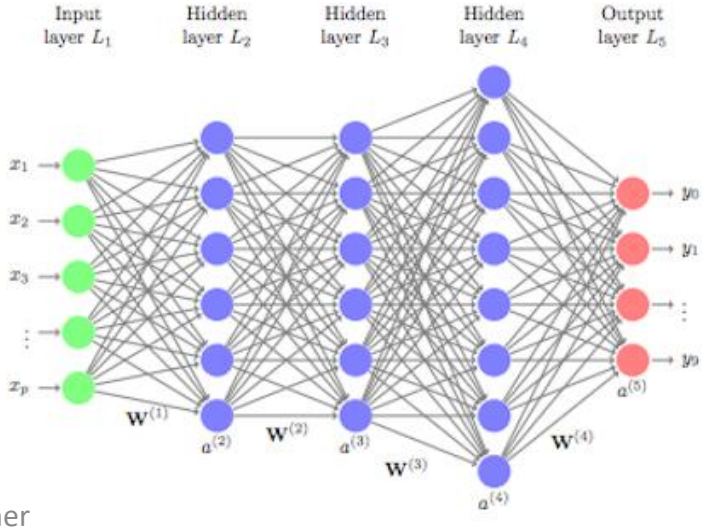
Lecture 2
Understanding Neural Networks

Dr. Magdi AMER

Feedforward Neural Networks

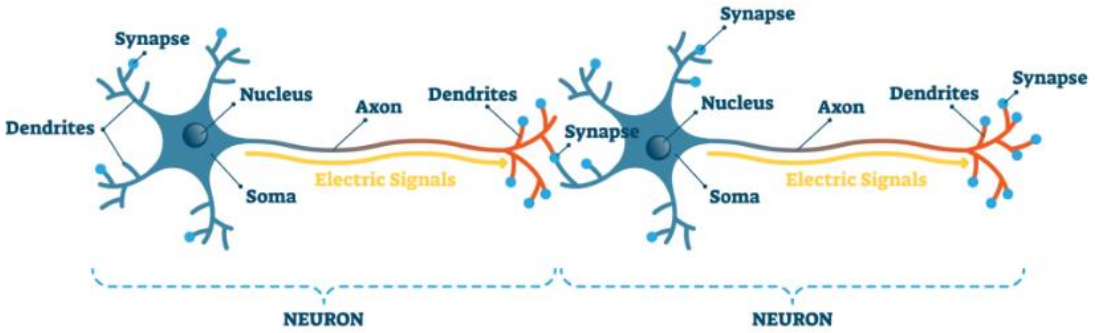
- There are many ways to design the NN
- One of the most common Feedforward
  - 1 layer for input
  - 1 layer for output
  - N layer (hidden) in the middle
- A neuron in layer N is connected to all neurons in N-1 and N+1, but not N





Training

- Training is the process of adjusting
  - Weight
  - Threshold



- This adjustment is done by calculating the error and using techniques to reduce the error

Training the Neural Network

- There are many ways to train the neural network
- We will discuss the most common approaches
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning

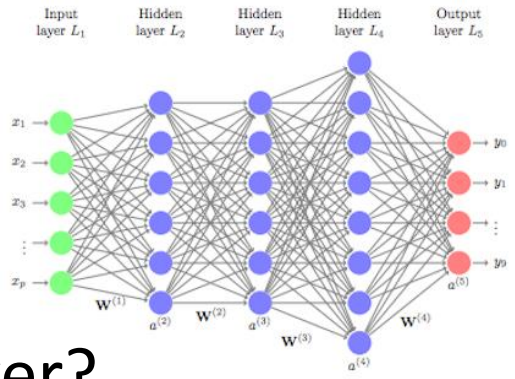
1) Supervised Learning

Overview

- To train a Neural Network to differentiate between images of cats and dogs:
  - Get a large number of cat images and dog images
  - Label each image
  - Divide into two packs
    - One for training
    - The other for validation
  - Use the first pack to train
  - Use the second to validate

Overview

- Drawback:
  - Hard to label all data
  - Sometimes I don't know how to label the data
- Design Decisions
  - How to model the input?
  - How many hidden layers?
  - How many neurons in hidden layer?
  - How much training data vs validation?
    - (50/50, 80/20, 90/10 ?)
  - What is the optimal value for the error?



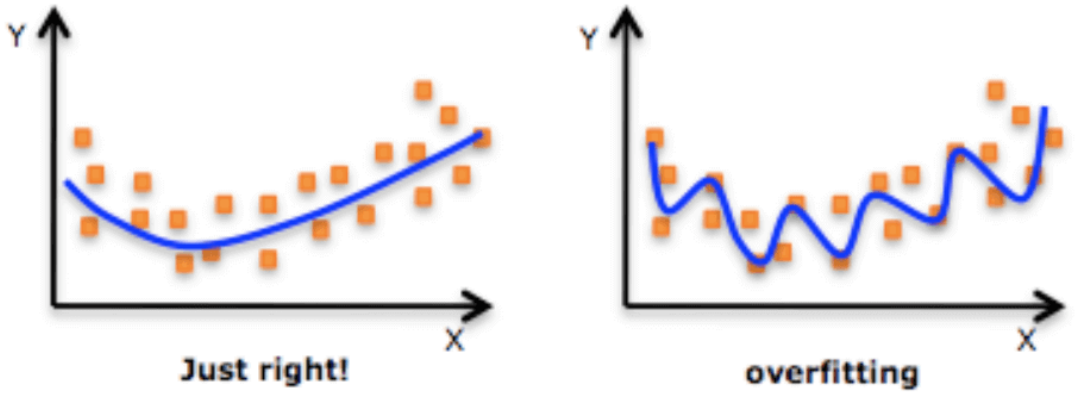
More on the Error

- Formula
  - Average of absolute
    - If error |1|, |-3|, |5|
      - Average = 3
  - RMS:
    - ((1 + 9 + 25)/3)^1/2 = 3.56

$$x_{rms} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + \ldots + x_n^2}{n}}$$

$$x_{rms} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (actual_i - ideal_i)^2}$$

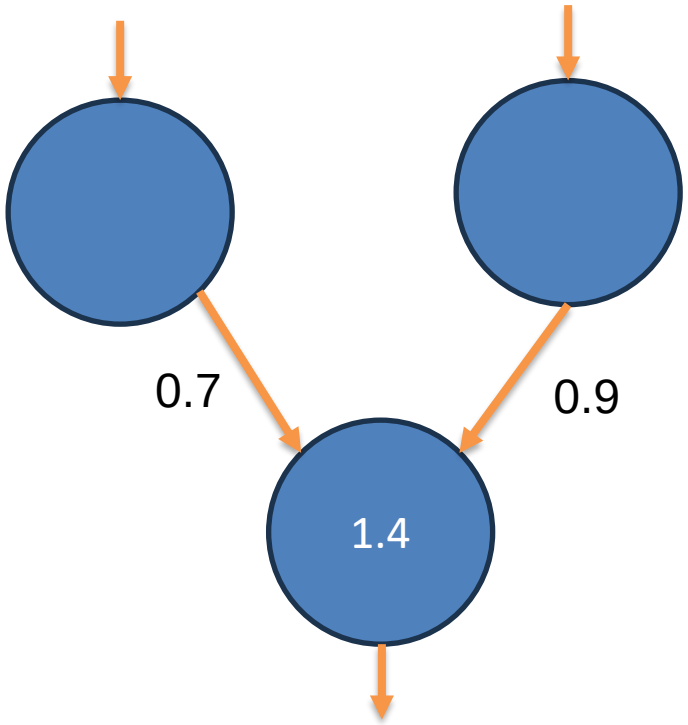
- Why we need error ➔ to prevent overfitting



1.1) Training *A*rtificial *N*eural *N*etwork (ANN)

Training an AND

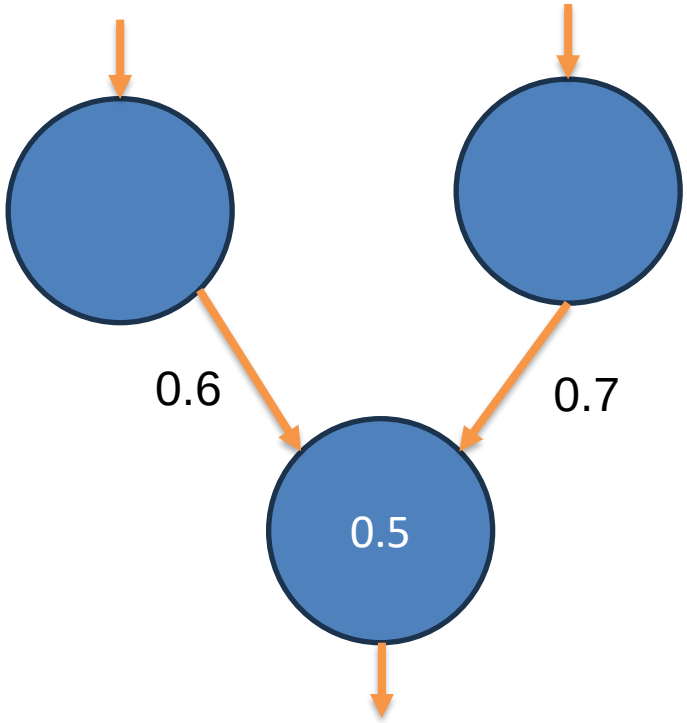
- Using Zero hidden layer



AND

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

Training an OR

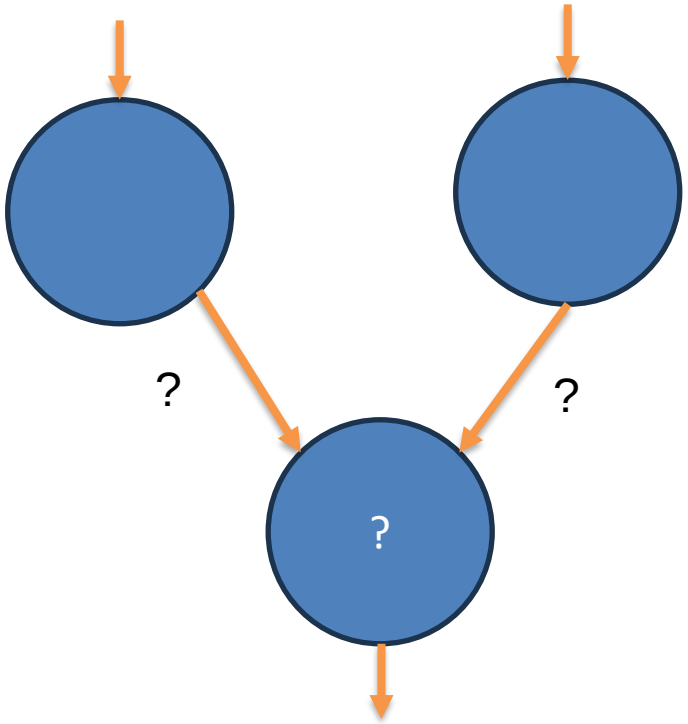


OR

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

Training an XOR

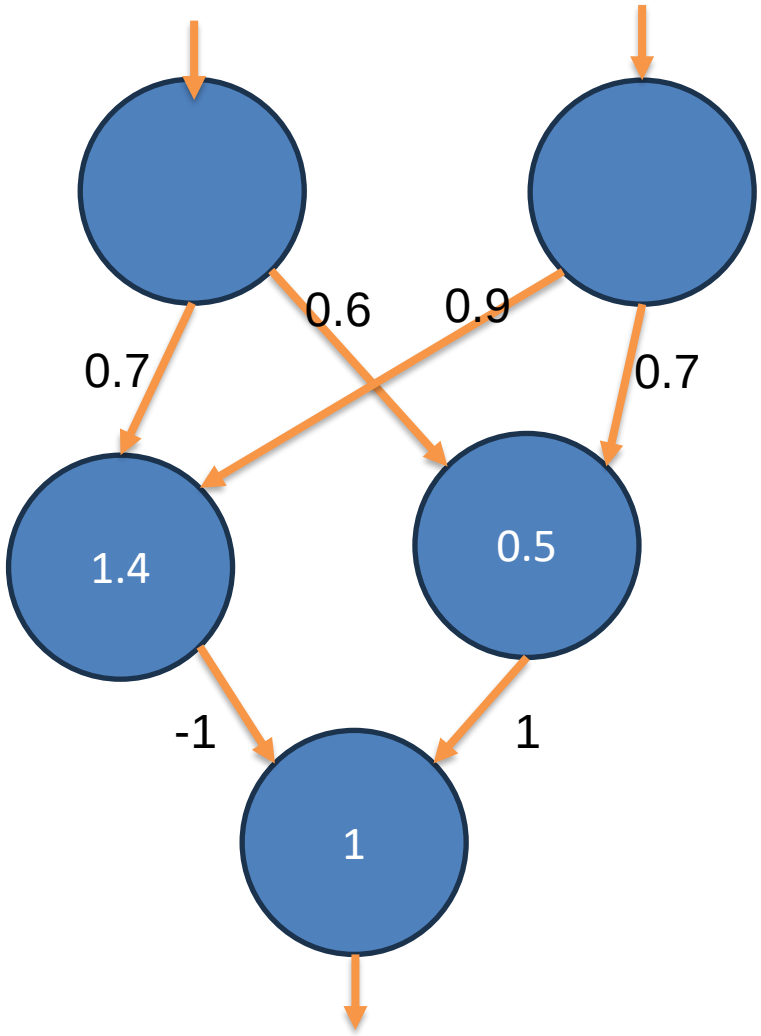
- Zero hidden layer will not be capable of representing this



XOR

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

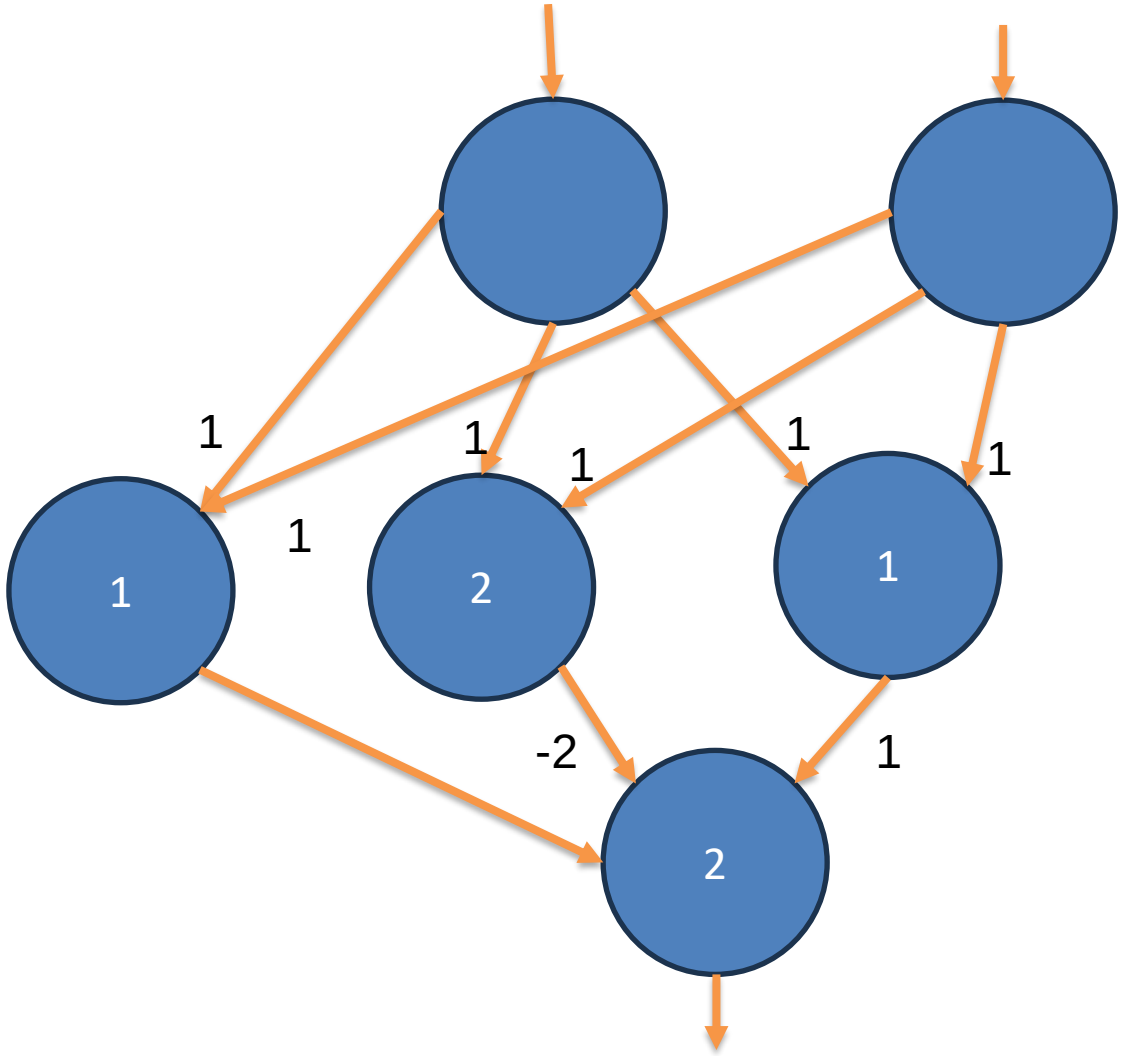
Training an XOR



XOR

x	*y*	$x \oplus y$
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

Training an XOR



XOR

x	*y*	$x \oplus y$
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

1.2) Activation Function

Calculating the Output

$$\sigma(w_1 a_1 + w_2 a_2 + \cdots + w_n a_n + b)$$

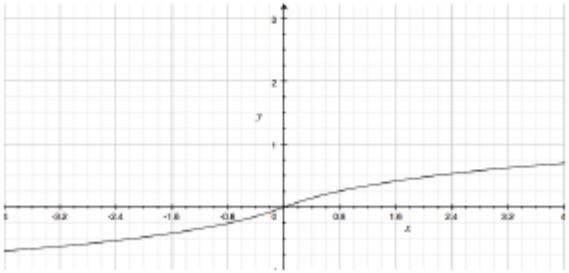
σ(x) is a function that takes as a parameter the sum of weights + bias

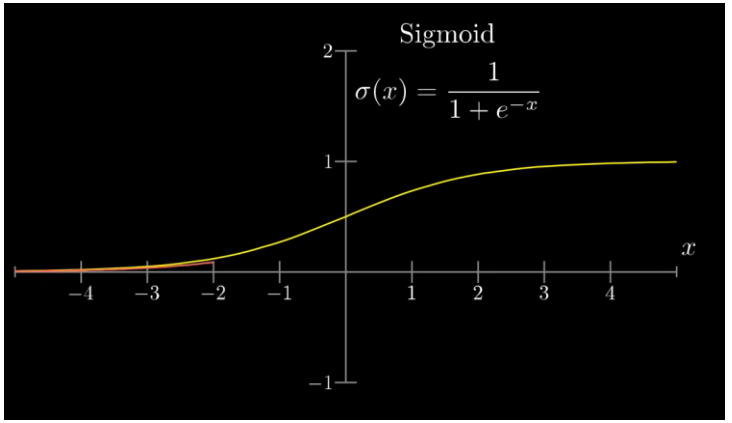
ANN with a non-linear function will be capable of solving more advanced problems than if it was using a linear function, but training will be much more difficult

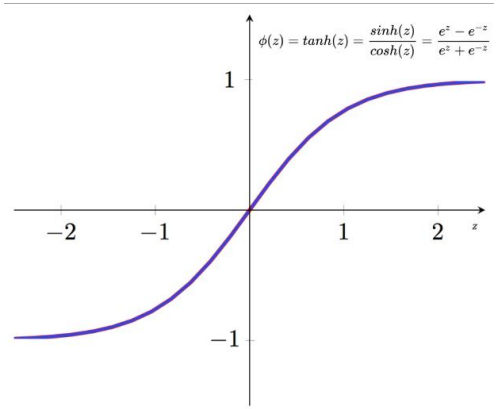
Activation Function

$$f(x) = \begin{cases} \log(1+x) & , x >= 0 \\ \log(1-x) & , \text{otherwise} \end{cases}$$

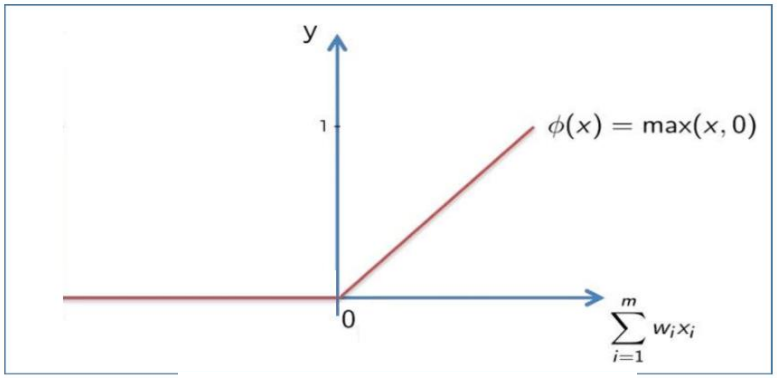
Logarithmic Activation Function







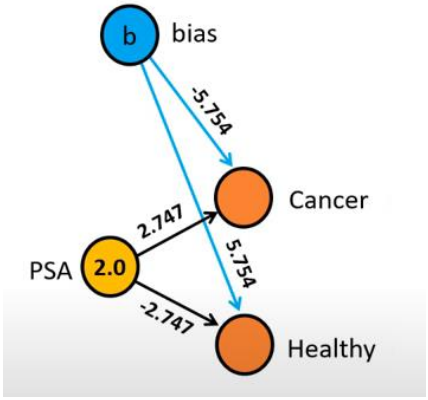
The Hyperbolic Tangent Function

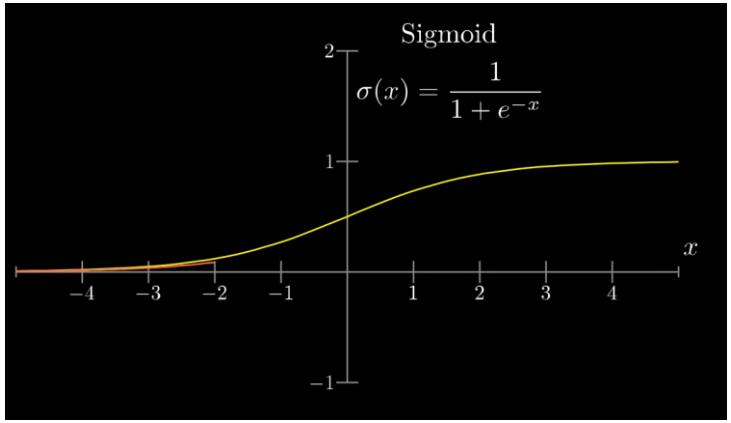


The Rectifier Function

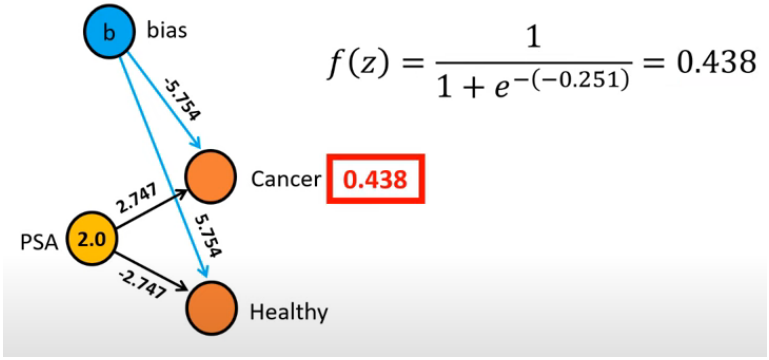
Activation Function Example

Status	PSA
Cancer	3.8
Cancer	3.4
Cancer	2.9
Cancer	2.8
Cancer	2.7
Cancer	2.1
Cancer	1.6
Healthy	2.5
Healthy	2.0
Healthy	1.7
Healthy	1.4
Healthy	1.2
Healthy	0.9
Healthy	0.8



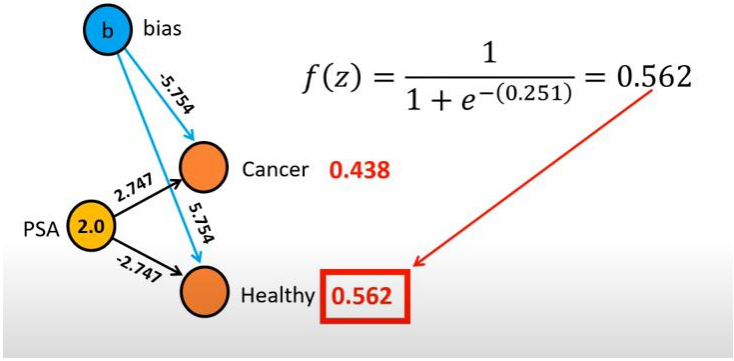


$$z = -5.754 + 2.747 \cdot 2.0 = -0.251$$



$$f(z) = \frac{1}{1+e^{-(-0.251)}} = 0.438$$

$$z = 5.754 + (-2.747) \cdot 2.0 = 0.251$$



$$f(z) = \frac{1}{1+e^{-(0.251)}} = 0.562$$

Calculating the Output

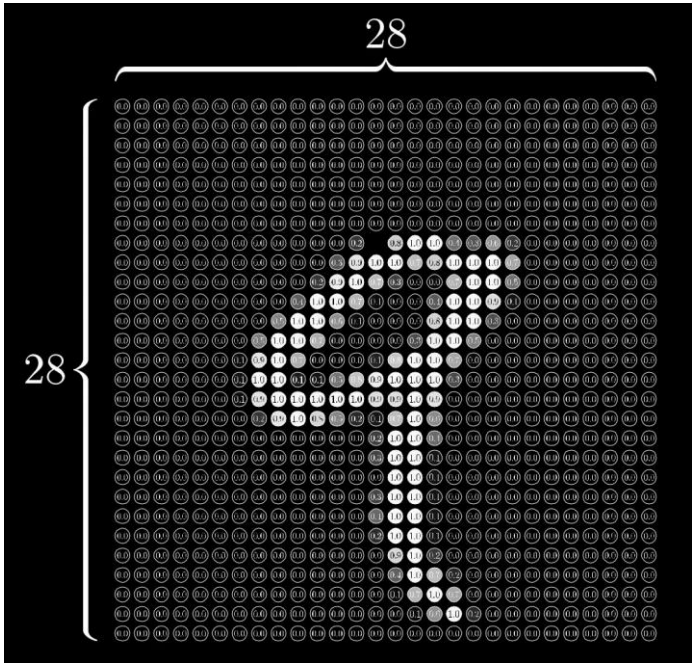
For a single neuron

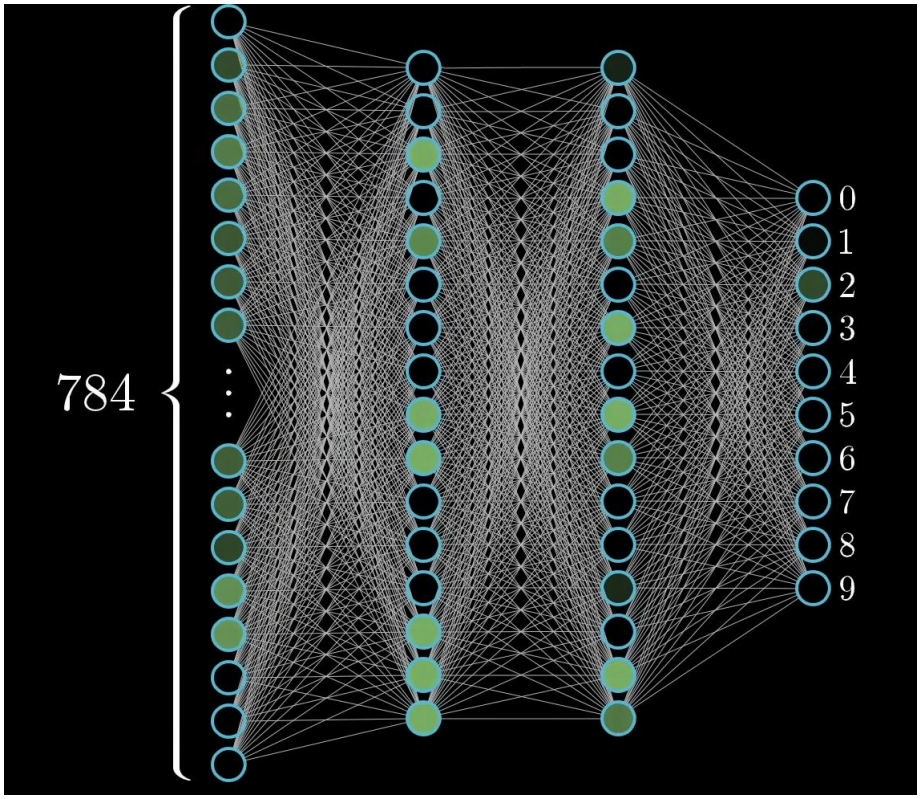
$$\sigma(w_1 a_1 + w_2 a_2 + \cdots + w_n a_n + b)$$

For n neurons

$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}\right)$$

Number Recognition





MNIST Database

LeCun, Cortes and Burges

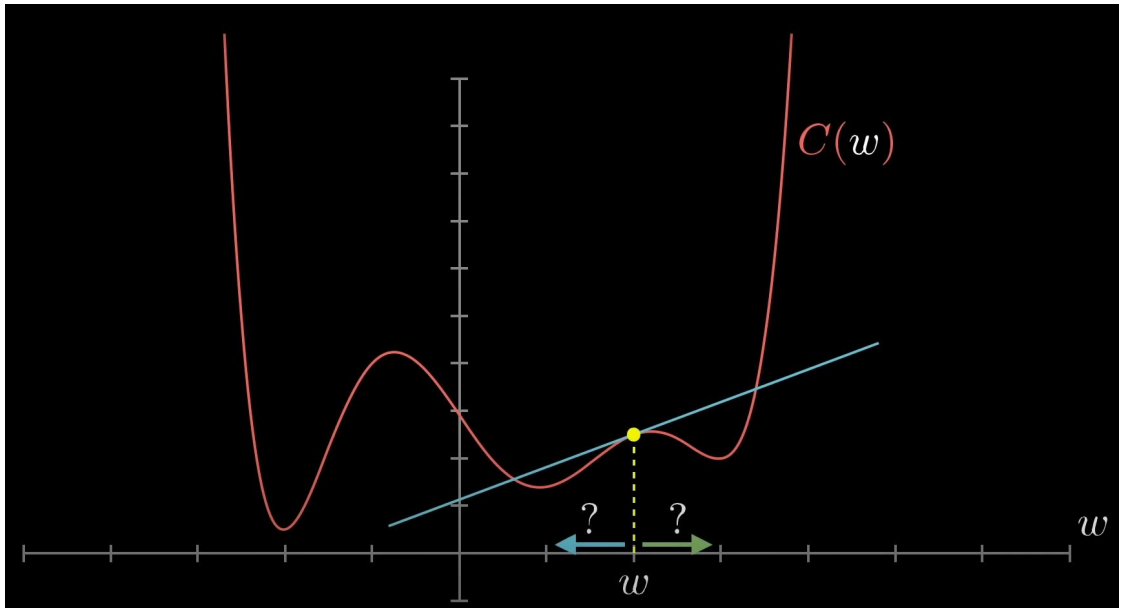
Number Recognition

$784 \times 16 + 16 \times 16 + 16 \times 10$
weights

$16 + 16 + 10$
biases

13,002

Learning $\rightarrow$ Finding the right weights and biases



Step function of the slope reduce the probability of overshooting or being too slow

1.3) Back Propagation

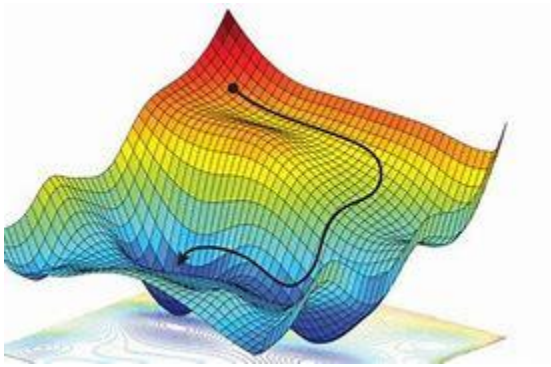
The output Node delta

- It represents how much the output of that node needs to change to minimize the network's overall error.
- It is equal to: (the derivative of the error function) X (The derivative of the activation function)
- When using the mean square error $\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$
  - It equal to $\delta_i = (\hat{y}_i - y_i)\phi_i'$

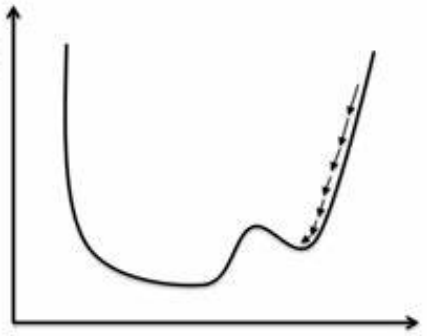
    For a sample i, it is equal to the optimal output minus the actual output * the derivative of the activation function

The gradient

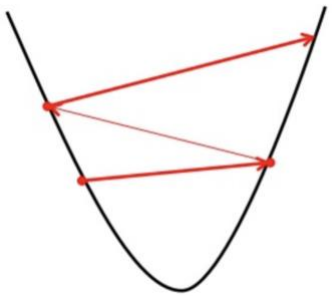
- The node delta is the gradient
- It is the change in the weight that we need to change to move toward the minimum



- Small step
  - Slow
  - Local minimum probability ↑



- Big step
  - May overshoot → diverge



Hidden Layer Delta

- The higher the weight → the higher the effect of change on the next layer
- Calculation of Delta for a Hidden Layer Node
  - Δ_j as the delta for a node j in a hidden layer,
  - w_{jk} as the weight from node j in the hidden layer to a node k in the layer ahead,
  - Δ_k as the delta for node k in the layer ahead, and
  - f' as the derivative of the activation function applied at node j.

  Then, the delta for node j is calculated as:

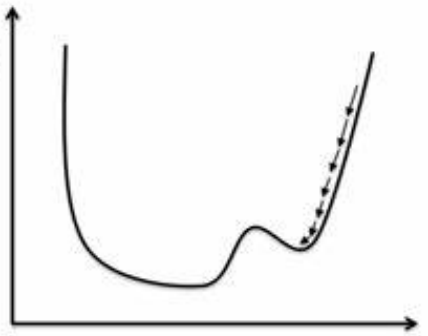
$$\Delta_j = \left(\sum_k w_{jk} \Delta_k \right) f'(z_j)$$

- This technique is called **Back Propagation**

Weight Update

Delta of output node: $\delta_i = (\hat{y}_i - y_i)\phi_i'$

Delta of hidden layer: $\Delta_j = \left(\sum_k w_{jk}\Delta_k\right) f'(z_j)$



Changes to the weight at any node

- η as the learning rate,
- μ as the momentum coefficient,
- Δ_j as the delta for node j in a hidden layer,
- o_i as the output from node i in the preceding layer,
- w_{ij} as the weight from node i to node j,
- $V_{ij}^{(old)}$ as the previous update velocity for weight w_{ij},

The change in the weight w_{ij}, including momentum, is now described by updating the velocity V_{ij} and then applying it to w_{ij}:

1. **Update the velocity** V_{ij} for the weight w_{ij}:

$$V_{ij}^{(new)} = \mu V_{ij}^{(old)} - \eta\Delta_j o_i$$

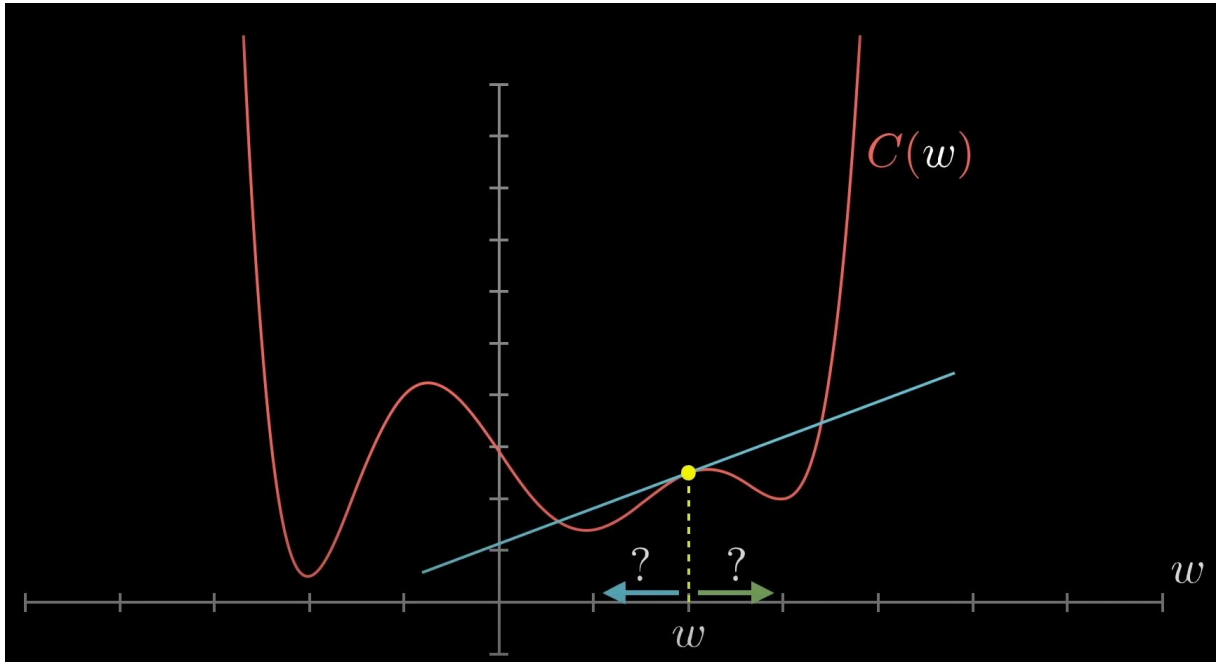
1. **Update the weight** w_{ij} using the new velocity:

$$w_{ij}^{(new)} = w_{ij}^{(old)} + V_{ij}^{(new)}$$

Understanding momentum

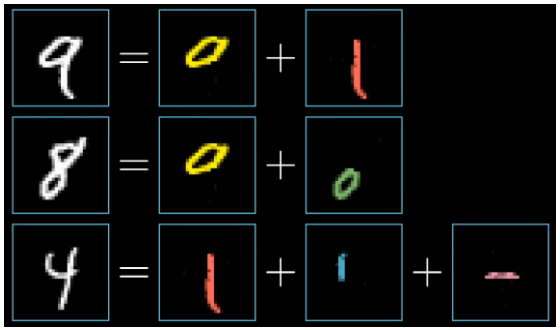
$$V_{ij}^{(new)} = \mu V_{ij}^{(old)} - \eta \Delta_j o_i$$

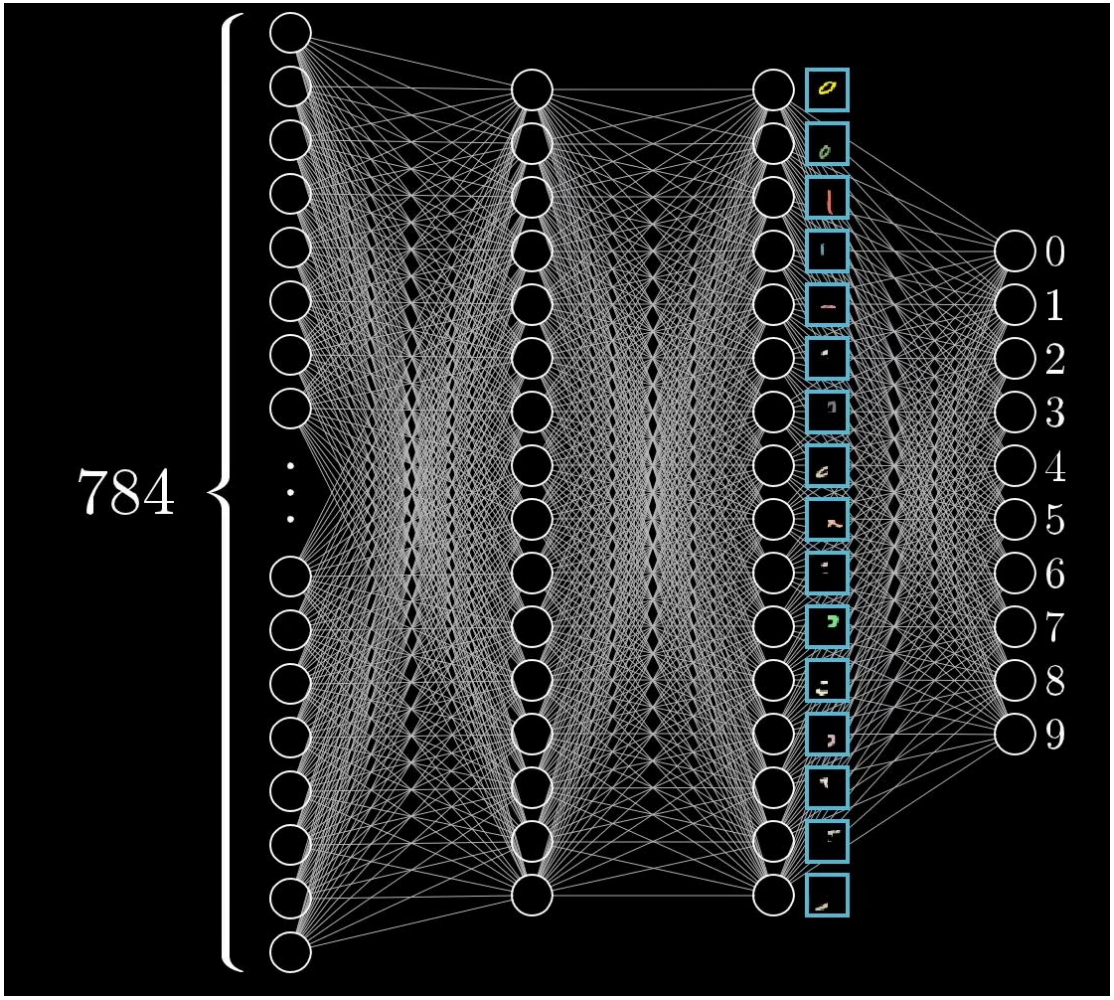
$$w_{ij}^{(new)} = w_{ij}^{(old)} + V_{ij}^{(new)}$$



1.4) Deep Neural Network

Deep Learning



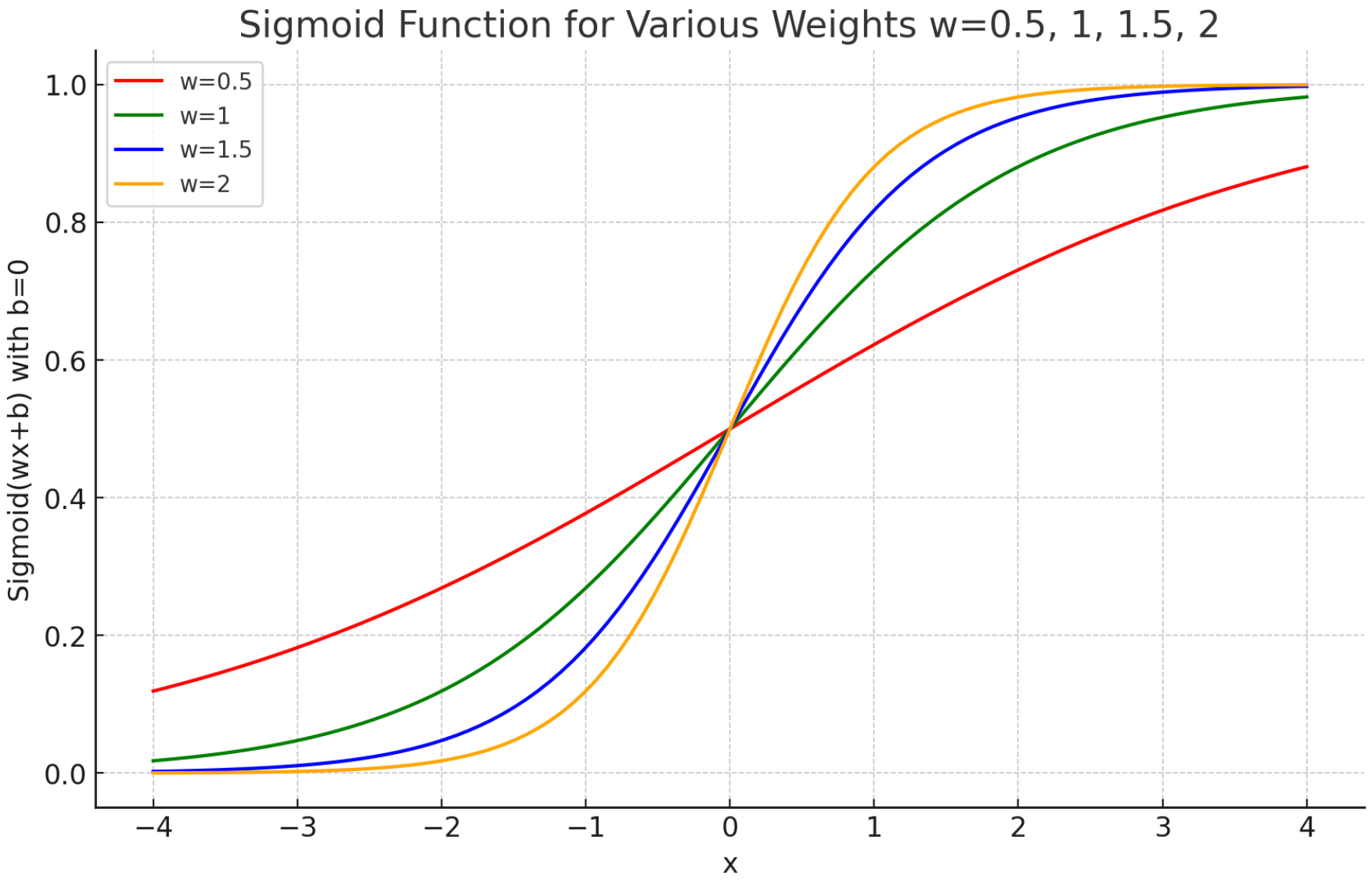


1.5) Why Bias is needed

Effect of the Weight on Sigmoid

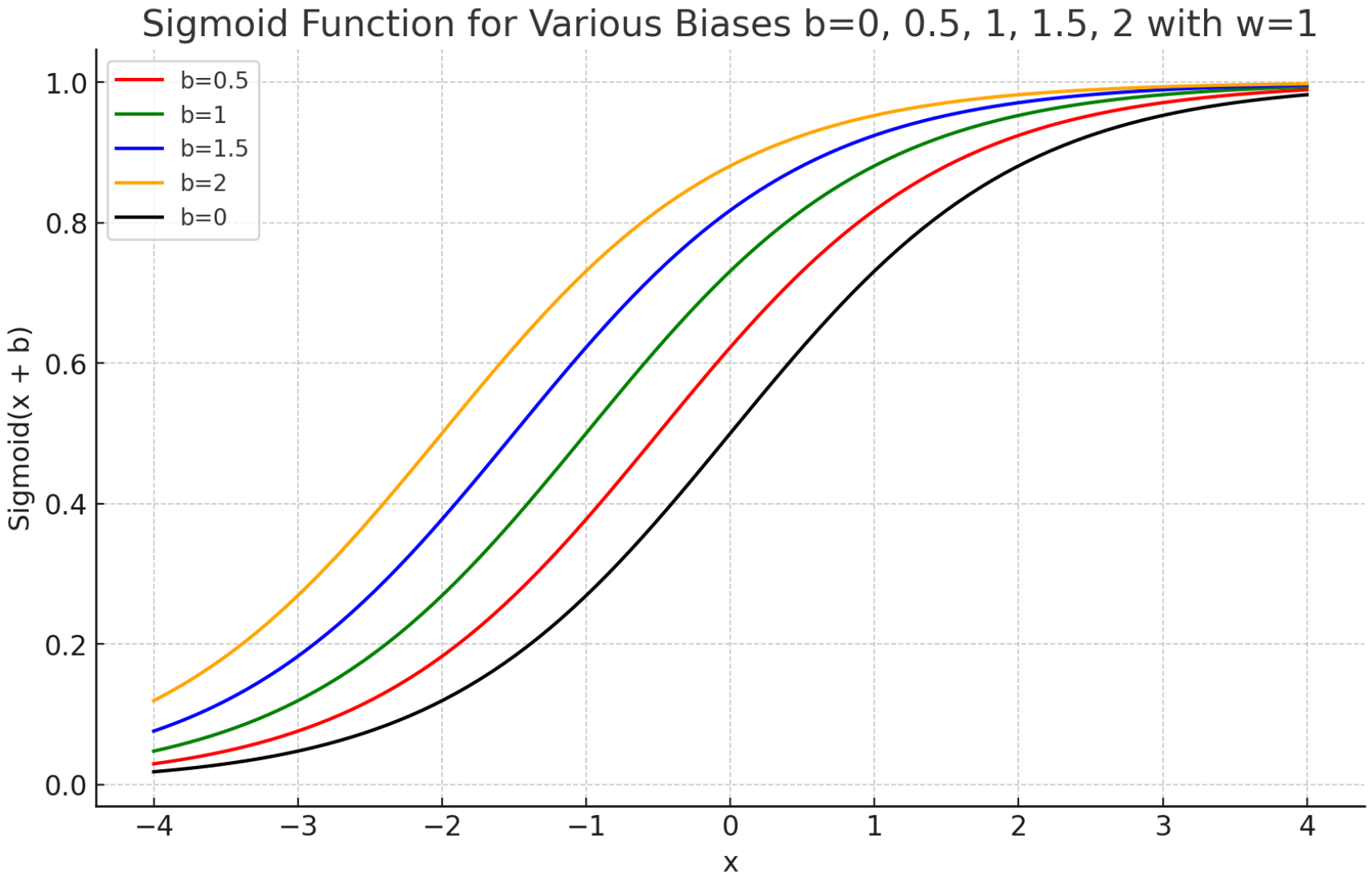
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\sigma(w_1 a_1 + w_2 a_2 + \cdots + w_n a_n + b)$$



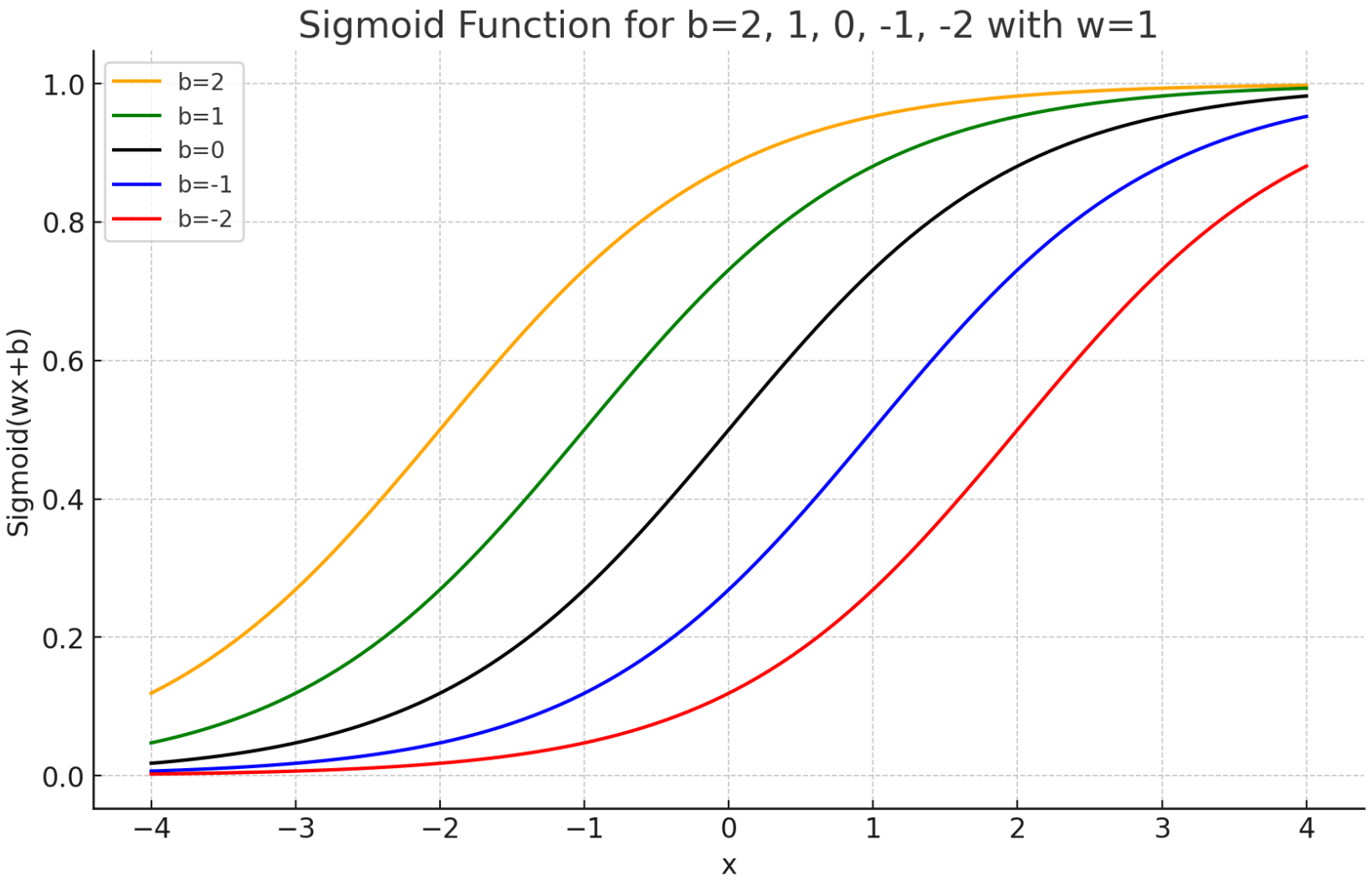
Effect of the Bias on Sigmoid

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



Effect of the Bias on Sigmoid

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

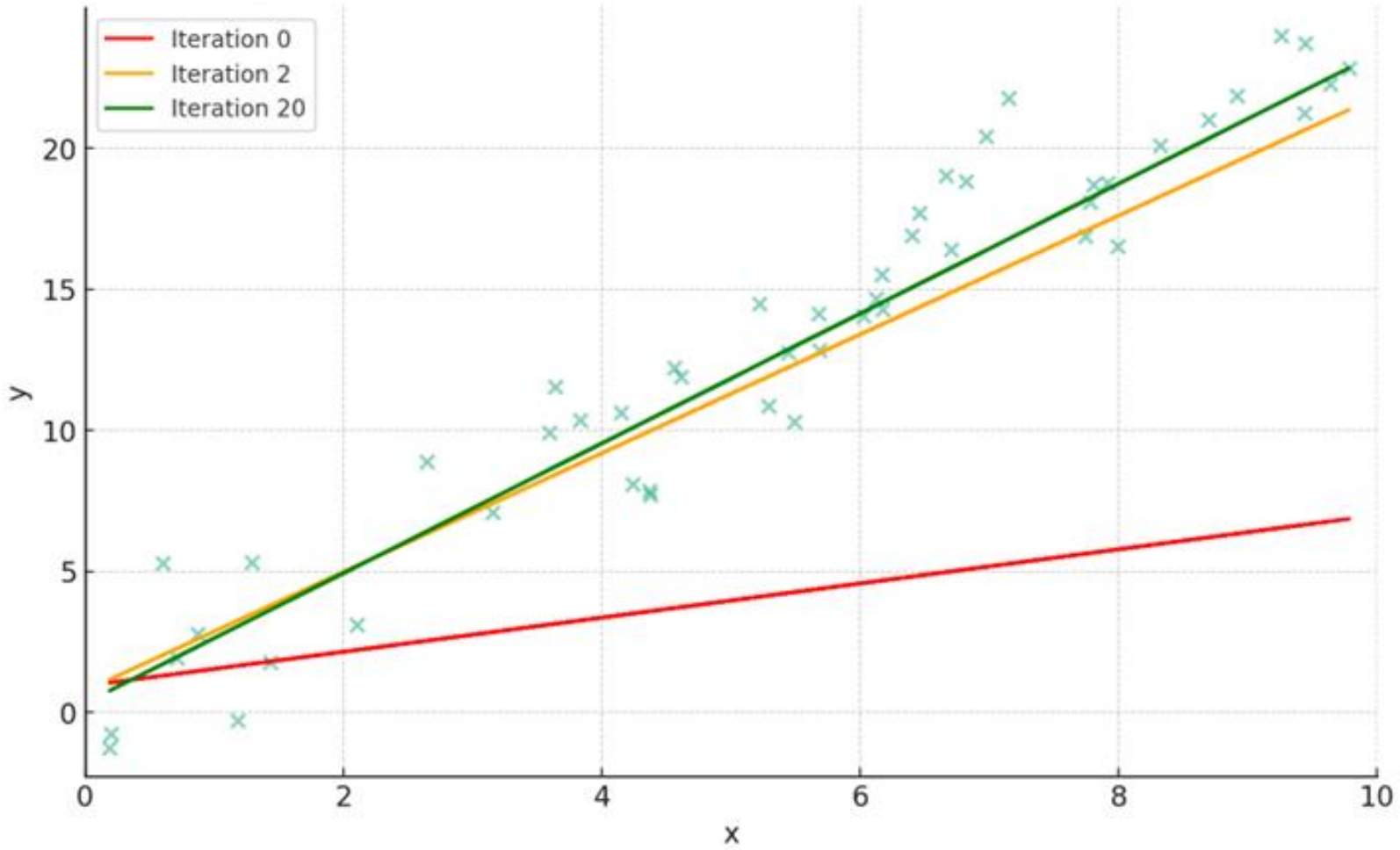


1.5) Classification and Regression

Regression

- Simple Linear Regression
  - estimates the linear relationship between
    - a response (also known as the dependent variable)
    - and one or more independent variables
  - The goal is to find a straight line that best represents the relationship between the response variable and independent variables.
- The goal of ANN when applied to regression is to predict the value of the dependent variable from a set of data points.
- Modern research recommend using ***linear activation function*** on output node for regression.

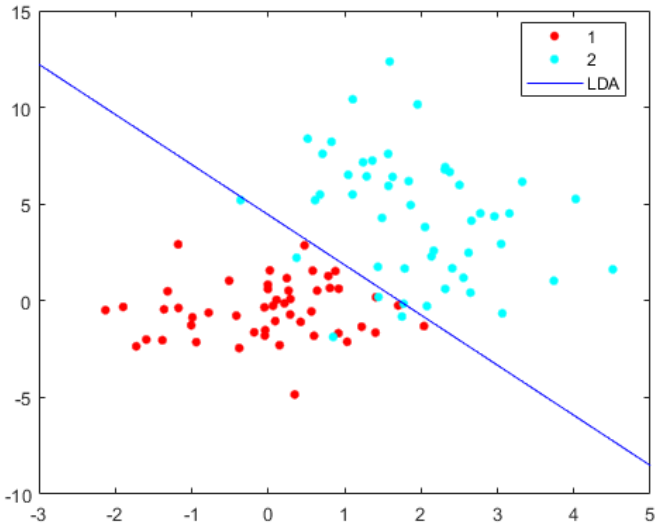
Regression



Classification

- Predicts
  - which category or class a new observation belongs to
  - based on a training set of data containing observations whose category membership is known.

- Softmax is the recommended activation function for classification for the output node.



Softmax

- the Softmax calculate the probability of an item be of a class X
- It is based on the output value of all neurons of the output layer

$$\phi_i(x) = \frac{exp(x_i)}{\sum_j exp(x_j)}$$

Softmax

- Assume an output of:
  - 0.5, 1.2, 1.1, 0.8
    - e^0.5 = 0.16
    - e^1.2 = 0.32
    - e^1.1 = 0.29
    - e^0.8 = 0.22
  - Sum = 10.2
  - Result
    - 0.5 ➔ 0.16
    - 1.2 ➔ 0.33
    - 1.1 ➔ 0.29
    - 0.8 ➔ 0.22

Total = 1

$$\phi_i(x) = \frac{exp(x_i)}{\sum_j exp(x_j)}$$

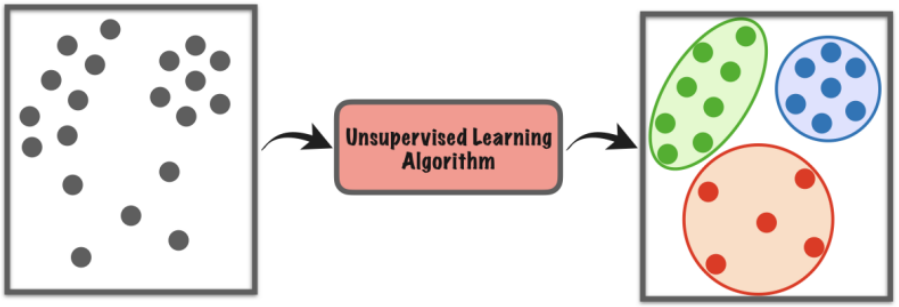
Activation Function Best Practice

- Since its introduction, researchers have rapidly adopted the rectified linear unit (ReLU) due to superior training results.
- As a result, most neural networks should utilize
  - ReLU on hidden layers
  - softmax on the output layer for classification.
  - linear on the output layer for regression.

2) Unsupervised Learning

Unsupervised Learning

- Best known application is clustering
- When dataset too big to label
- When we don't know how to label
  - Example Recommendation of movies
- One of the most used algorithm is ***K-Means***



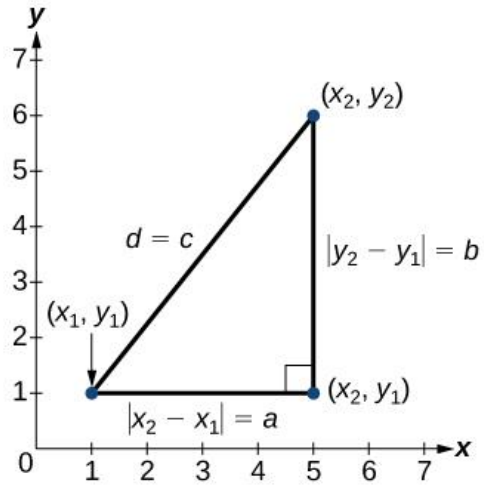
Calculating the distance

For ***2*** dimension

$$c^2 = a^2 + b^2 \rightarrow c = \sqrt{a^2 + b^2}$$

$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$$

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



For ***n*** dimension

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

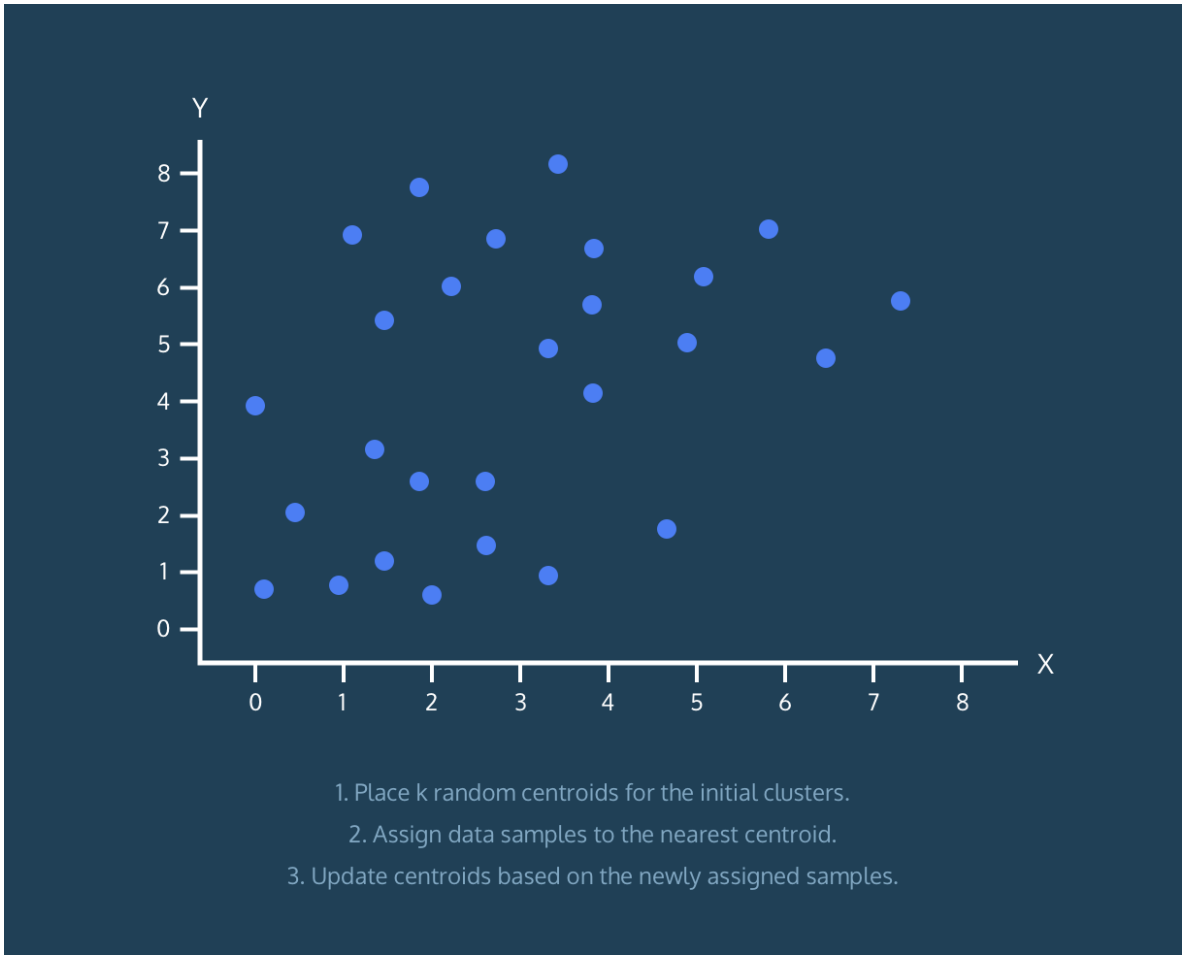
K-Means Algorithm

- 1) Initialization: For K clusters
  - Two techniques
    - Forgy Initialization:
      - each point is assigned randomly to a cluster
      - Centroids are calculating from these points.
      - Centroid is:
        - » The geometric center or center of figure
        - » It is calculated by the arithmetic mean position of all the points in the surface of the figure
    - K-Means ++
      - We start with K centroids randomly chosen
      - A point is assigned to the cluster that is the nearest to it

K-Means Algorithm

- 2) Iteration
  - Calculate the new Centroids
  - Assign points to the nearest centroid
    - i.e., reduce distance
- 3) Repeat till any of these condition occurs
  - The centroids of new clusters do not change
  - Points stay in the same cluster
  - The maximum number of iterations is reached

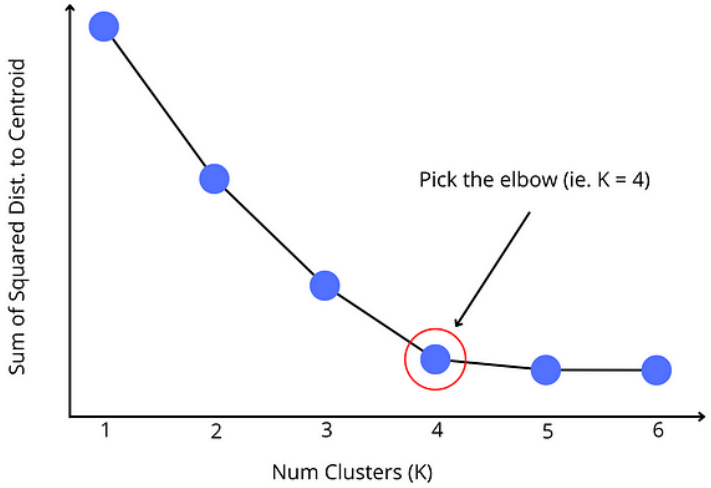
K-Means Example



Source: https://www.codecademy.com/learn/dspath-unsupervised/modules/dspath-clustering/cheatsheet

Finding the Number of Clusters

- Solve for k=1, k=2, …
  - Stop when Δ change is small
  - i.e. when slope converge



3) Reinforcement Learning

Algorithm

- When the system does a successful step ➔ +1
- When the system does a bad step ➔ -1
- Iterate over the problem multiple times till the result is satisfactory
- This is how
  - Robot learn to walk
  - Self driving care learn how to drive
  - Computer learn how to play any game by playing it

Advantages

- No labeling of data
- Innovative
  - Can design new solutions that were never even considered by humans.
- Goal oriented
  - Efficient when looking for a sequence of actions
  - Supervised and unsupervised learning are more suitable for input ➔ output problems
- Dynamic
  - It continuously learn without having to re-train and re-deploy