Neural Network

Course Outline

- Introduction
- Biological Neural Networks
- Perceptron, Multilayer and Recursive Nets
- Gradient Descent
- Back Propagation

Introduction

- Neural networks are a subset of machine learning algorithms inspired by the structure and functioning of the human brain.
- ***They are designed to recognize patterns and learn from data, allowing computers to perform tasks without being explicitly programmed for them.***
- A neural network consists of interconnected nodes, called neurons, organized in layers.
- The most common type of neural network is the feedforward neural network, where information flows in one direction, from the input layer through the hidden layers to the output layer.

Introduction

- Each neuron in a neural network receives input signals, performs a computation, and produces an output signal.
- The computation involves applying weights to the inputs, summing them up, and passing the result through an activation function.
- The ***activation function introduces non-linearity to the network***, enabling it to learn complex patterns.
- During the training process, the neural network adjusts the weights of its connections based on the input data and the desired output.
- This adjustment is typically done using optimization algorithms like ***backpropagation,*** which calculates the gradient of the network's error with respect to its weights and updates them accordingly.

Introduction

- The goal is to minimize the difference between ***the predicted output and the true output.***
- Neural networks can be used for various tasks, including classification, regression, and pattern recognition.
- They have achieved significant success in areas such as computer vision, natural language processing, speech recognition, and recommendation systems.
- Deep learning, a subfield of machine learning, focuses on neural networks with multiple hidden layers, allowing them to learn more complex representations and solve more intricate problems.

Human Brain

- *Computers and the Brain: A Contrast*
- Arithmetic: 1 brain = 1/10 pocket calculator
- Vision: 1 brain = 1000 super computers
- Memory of arbitrary details: computer wins
- Memory of real-world facts: brain wins
- A computer must be programmed

Computer Operations

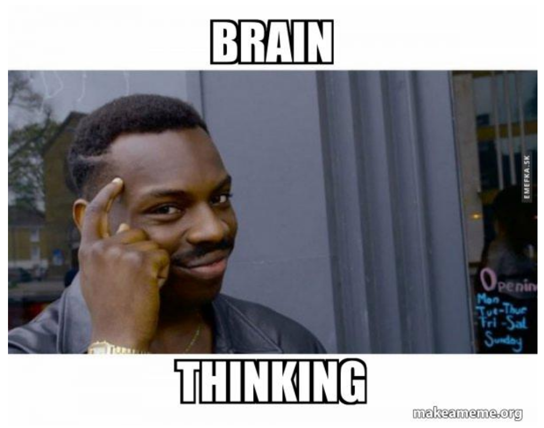
Traditionally computers execute a sequence of instructions to accomplish a set task

- **This is a powerful technique if you know the 'algorithm'**
- **It's not very useful if you don't !!**

There are many interesting tasks where the algorithm is either unknown or unclear

- **Recognizing handwriting – Pattern recognition**
- **Playing table tennis – Interacting with the environment**
- **Balancing activities – Optimization**

The Question



- Humans find these tasks relatively simple
- We learn by example
- The brain is responsible for our ‘computing’ power
- If a machine were constructed using the fundamental building blocks found in the brain could it *learn* to do ‘difficult’ tasks ???

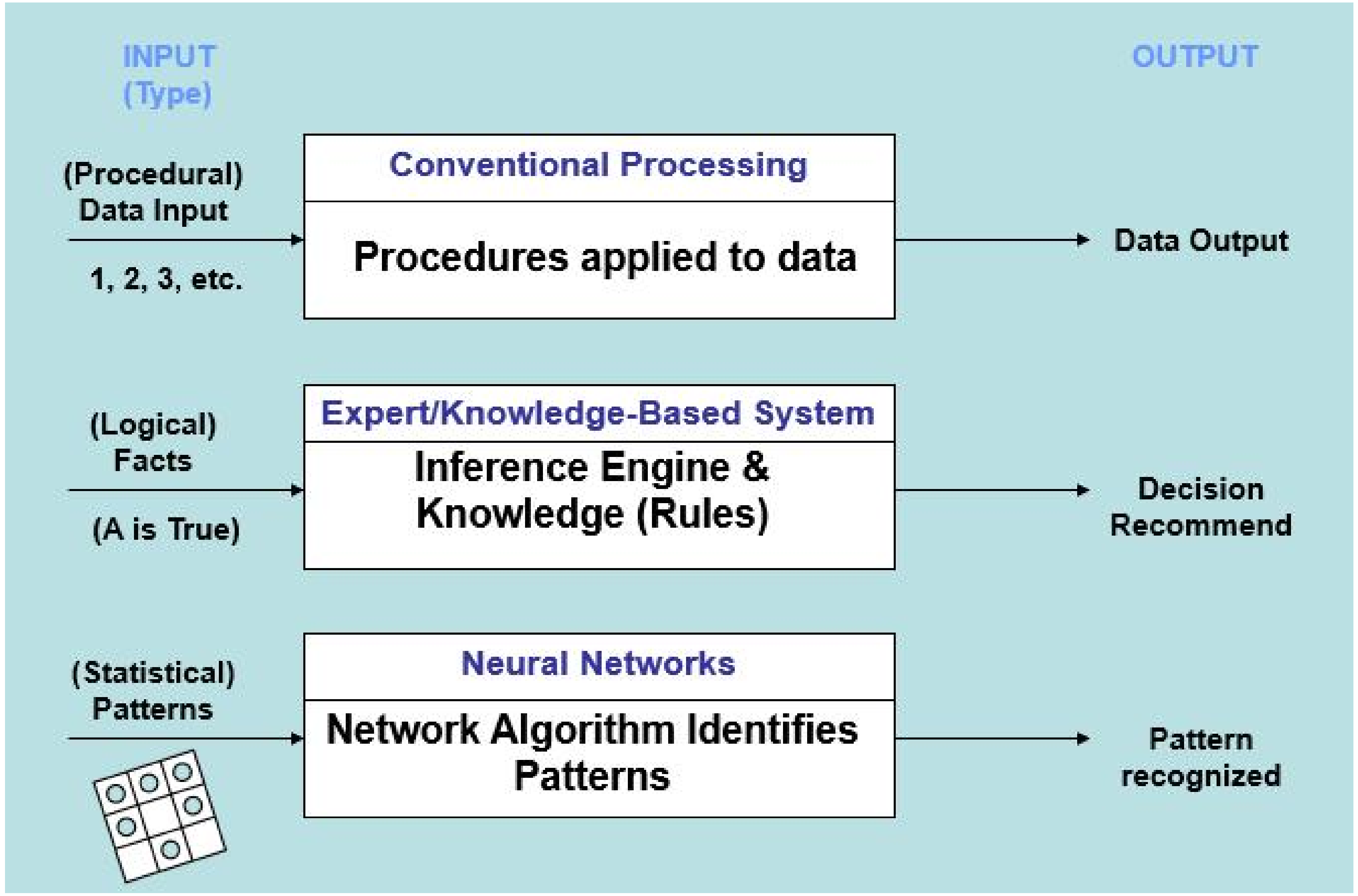
Definition

“. . . Neural nets are basically mathematical models of information processing . . .”

“. . . (neural nets) refer to machines that have a structure that, at some level, reflects what is known of the structure of the brain . . .”

“A neural network is a massively parallel distributed processor . . . “

Process Comparisons



Biological Neurons

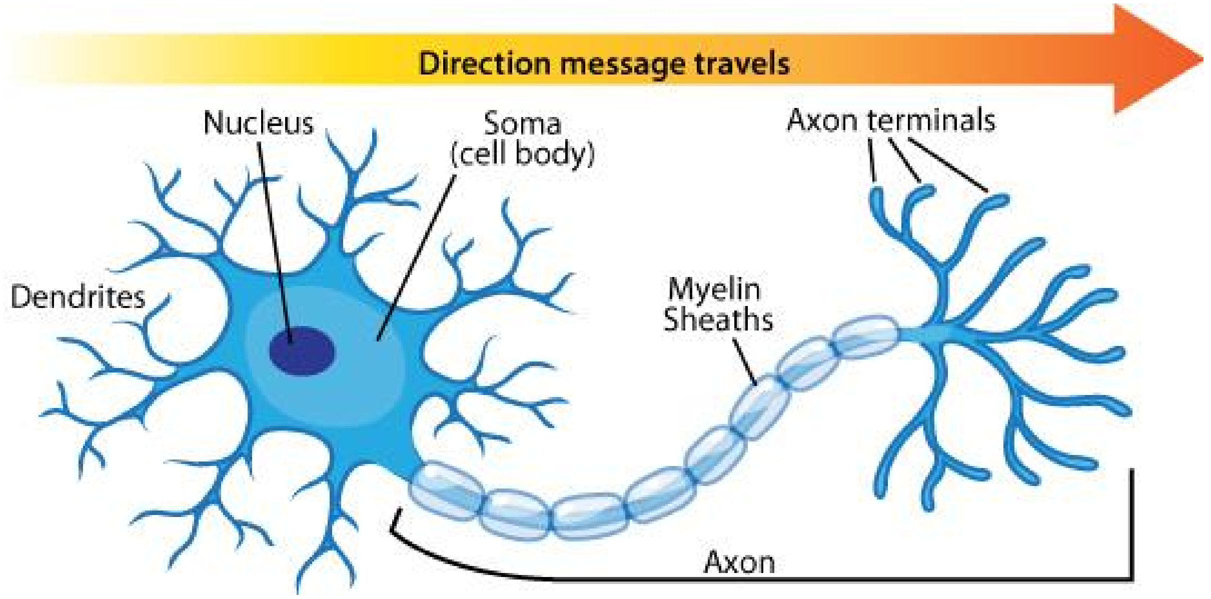
A neuron consists of two main parts:

Axon

- one per neuron
- excites up to 10^4 other neurons
- all or nothing output signal

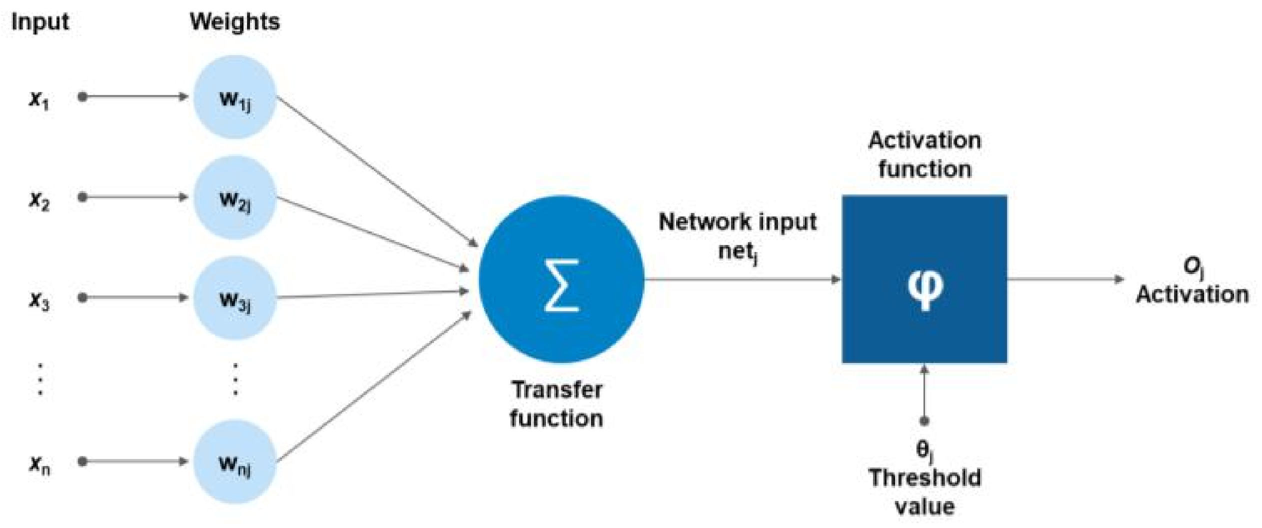
Dendrites

- 1 to 10^4 per neuron



Definition

- Each neuron may have over a thousand synapses
- Some cells in cerebral cortex may have 200,000 connections
- Total number of connections in the brain "network" is astronomical—greater than the number of particles in known universe.
- An artificial neuror
order characteristi



Applications

- **Diagnosis**
- Closest to pure concept learning and classification
- Some ANNs can be post-processed to produce probabilistic diagnoses
- **Prediction and Monitoring**
- Predict a continuation of (typically numerical) data
- **Decision Support Systems**
- recommender systems
- Provide assistance to human "subject matter" experts in making decisions
- Design (manufacturing, engineering), Therapy (medicine)
- Crisis management (medical, economic, military, computer security)
- **Control Automation**
- Mobile robots, Autonomic sensors and actuators

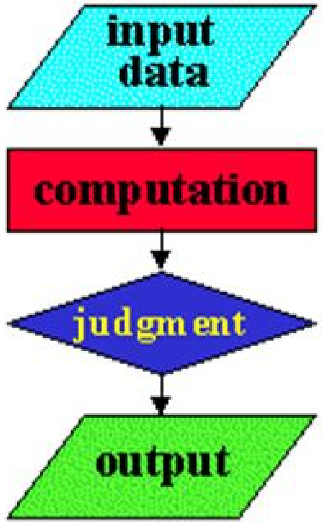
Advantage of Brain

Inherent Advantages of the Brain:

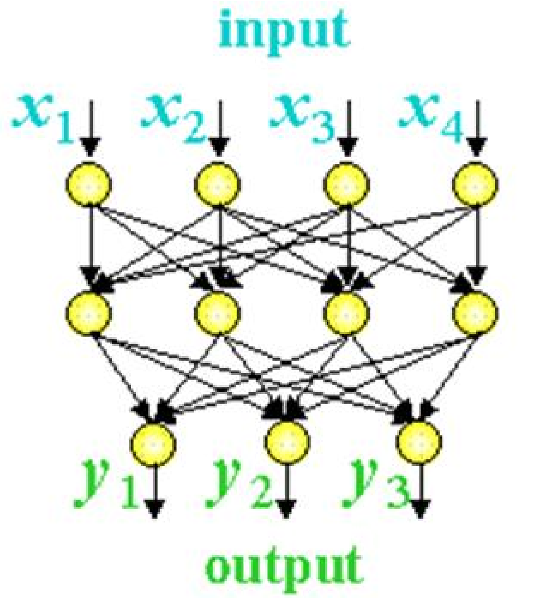
- "distributed processing and representation"
- Parallel processing speeds
- Fault tolerance
- Graceful degradation
- Ability to generalize

Neural Net Characteristics

- Massive Parallel Processing



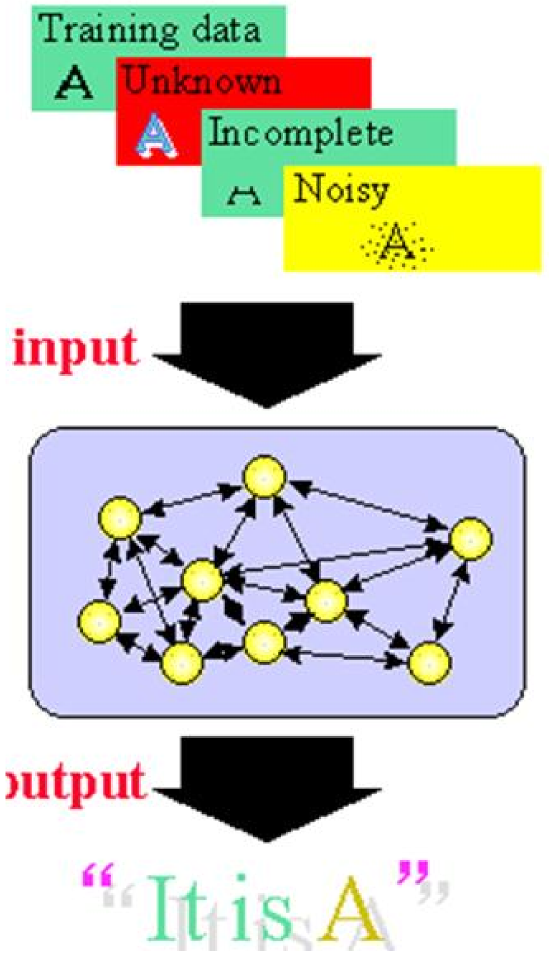
Sequential Information Processing (Computer)



Parallel Information Processing (Neural Network)

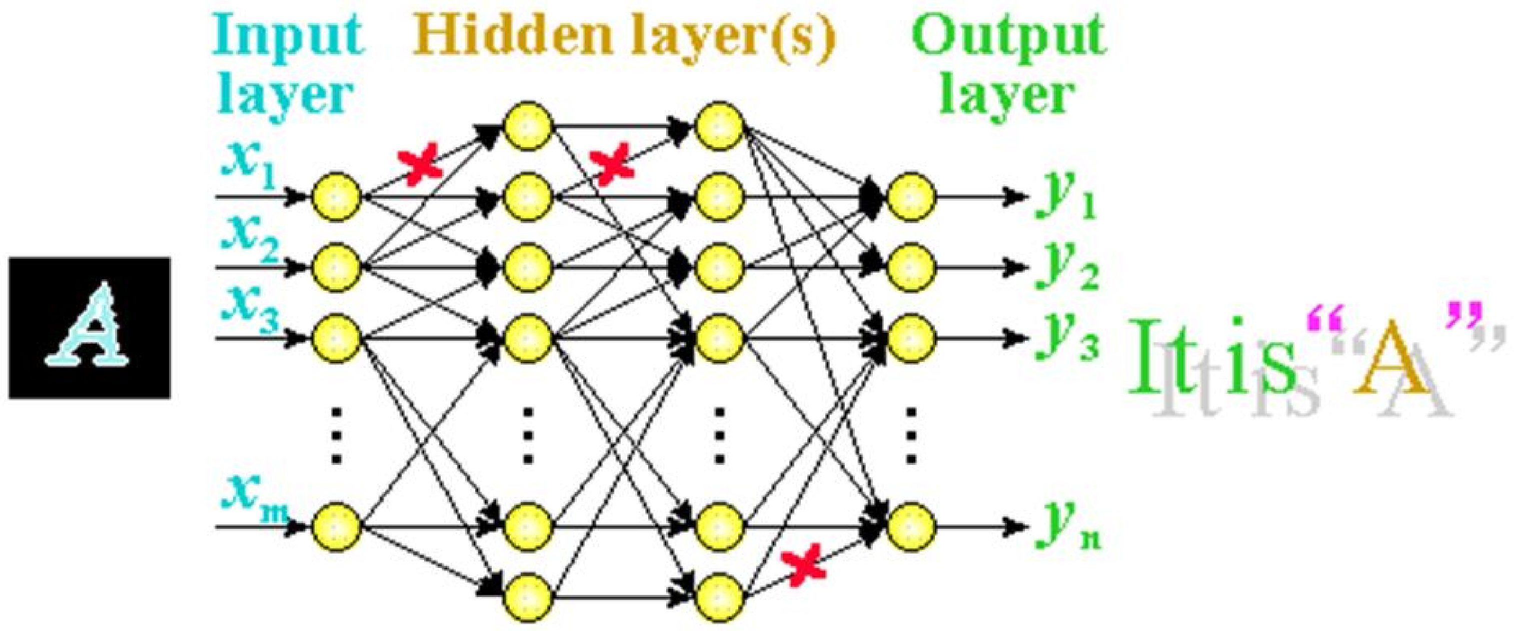
Neural Net Characteristics

- Once a neural net is trained, it can handle unknown, incomplete or noisy data



Neural Net Characteristics

- Unlike computer memory, information is distributed over the entire network, so a failure in one part of the network does not prevent the system for producing the correct output



History of Artificial Neurons

1943

Warren McCulloch and Walter Pitts propose the first artificial neuron.

1957

Frank Rosenblatt creates the first artificial neural network, the perceptron.

1969

Marvin Minsky and Seymour Papert publish Perceptrons, which shows that perceptrons are limited in their ability to learn.

1982

David Rumelhart, Geoffrey Hinton, and Ronald Williams develop the backpropagation algorithm, which allows neural networks to learn more complex tasks.

History of Artificial Neurons

1989

Yann LeCun develops the convolutional neural network, which is now one of the most widely used types of neural networks.

2006

Geoffrey Hinton, Terrence Sejnowski, and others develop the deep belief network, which is a type of neural network that can learn to represent data in a hierarchical manner.

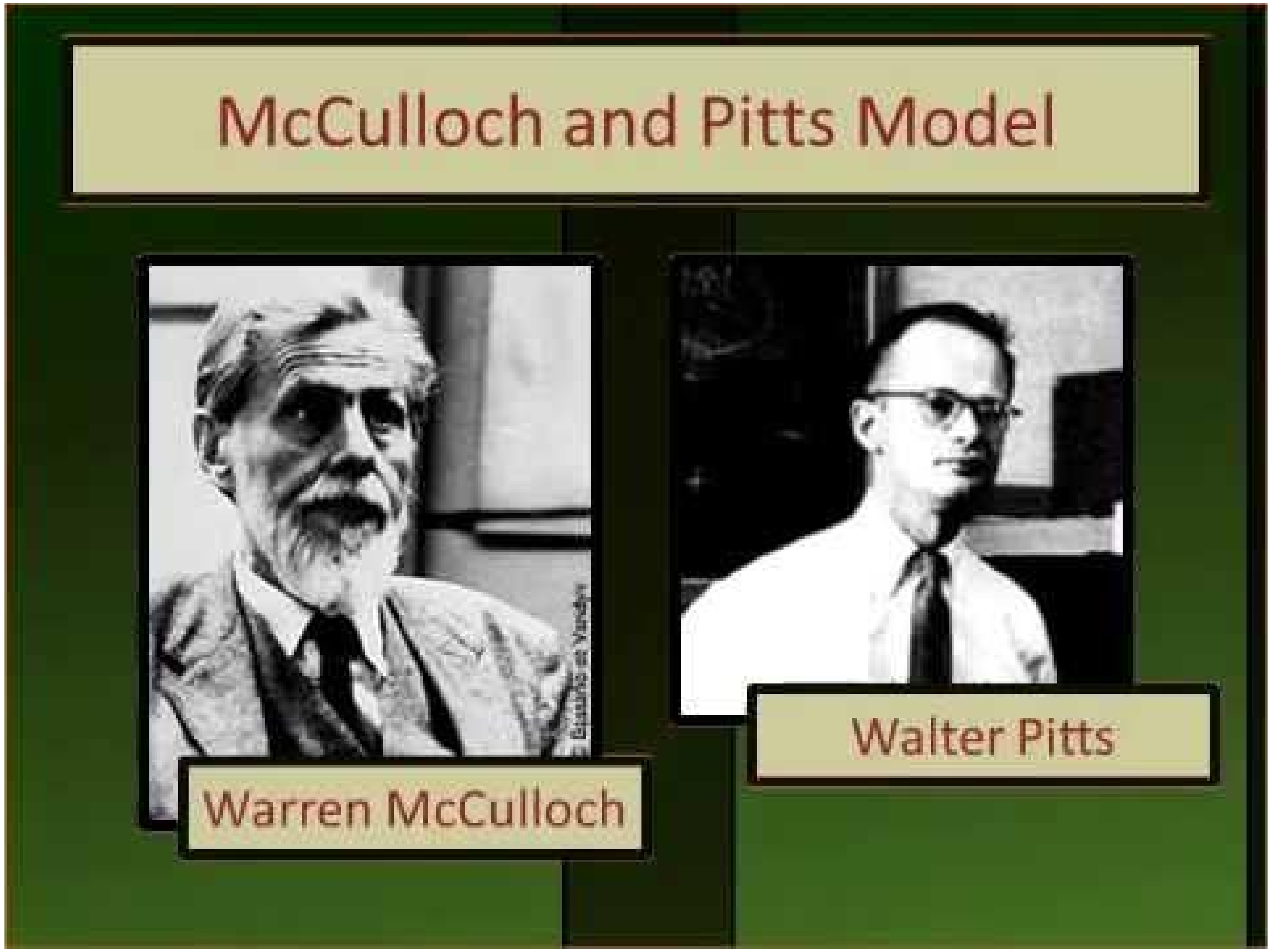
2012

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton develop the AlexNet convolutional neural network, which achieves state-of-the-art results on the ImageNet image classification task.

2015

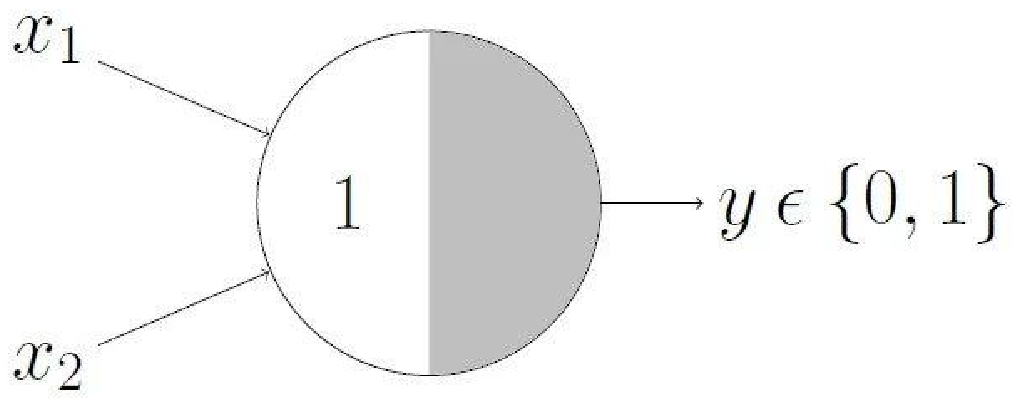
Google's AlphaGo program defeats a professional Go player, which is considered to be a major breakthrough for artificial intelligence.

McCulloch and Pitts Model



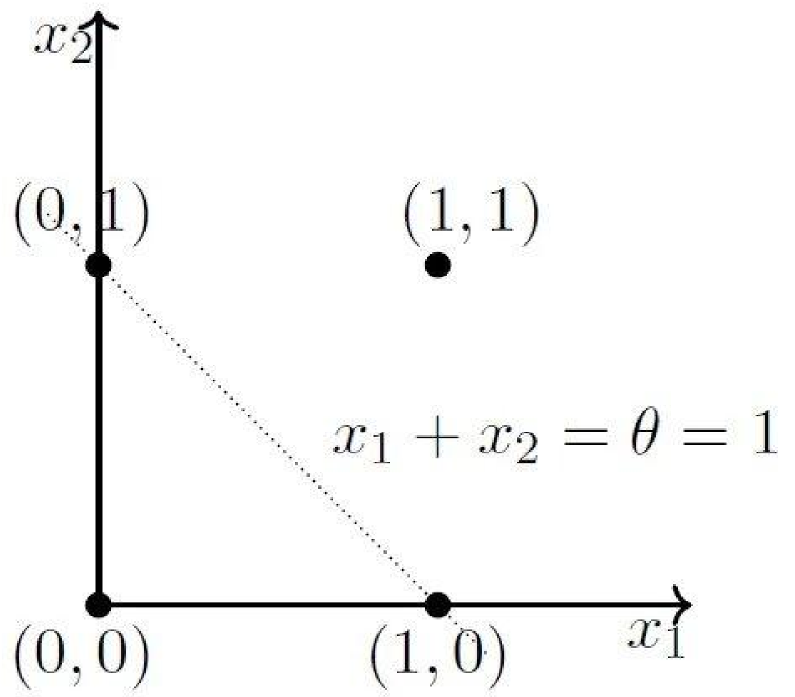
Warren McCulloch and Walter Pitts

- Warren McCulloch and Walter Pitts developed a mathematical model of a neuron in 1943.
- Their model, which is known as the McCulloch-Pitts neuron, is a simplified version of a biological neuron.
- It consists of a soma, dendrites, and an axon.
- The soma is the cell body, and the dendrites are the input pathways. The axon is the output pathway.
- The McCulloch-Pitts neuron is activated when the sum of its inputs exceeds a threshold.
- When the neuron is activated, it sends an output signal down its axon.

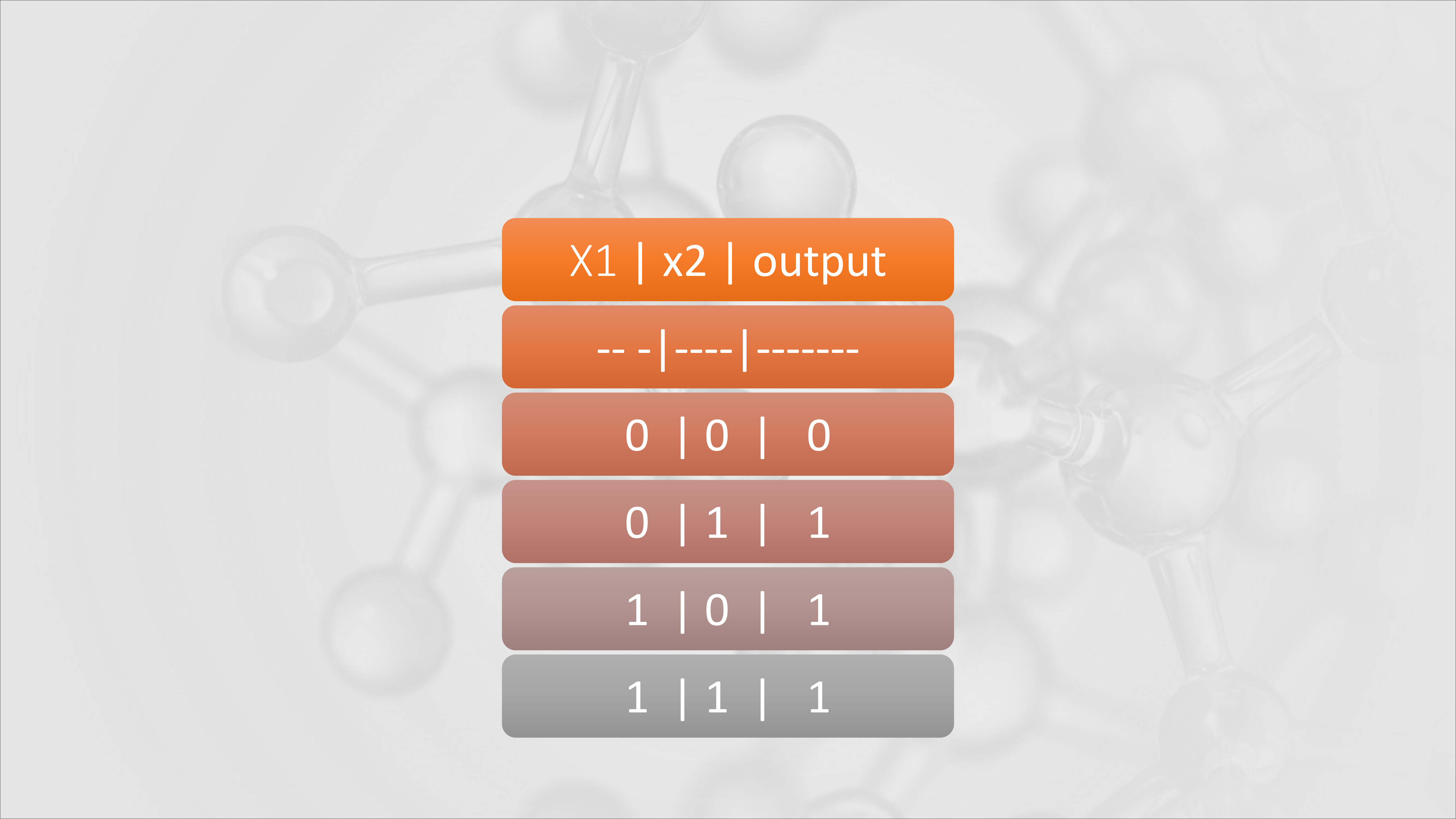


$OR\ function$

$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 1$$



- This neuron model, commonly known as the threshold neuron, is one of the earliest neural network models that were implemented.
- It has a binary output, where the output is either 1 (fired) or 0 (not fired).
- Each input is multiplied by a corresponding weight, which can take values between -1 and +1.
- The neuron also has a threshold value, denoted as T.
- The neuron fires if the sum of the inputs (x1 + x2) is greater than the threshold value T.
- In other words, if x1 + x2 > T, the neuron outputs 1; otherwise, it outputs 0.



X1	x2	output
0	0	0
0	1	1
1	0	1
1	1	1

To implement this using the threshold neuron model, we can assign the following values:

Threshold (T) = 0.5

Now, let's compute the output for each input combination:

- **When x1 = 0 and x2 = 0:**

  weighted sum = 0 + 0 = 0

  Since 0 <= 0.5, the output is 0.

- **When x1 = 0 and x2 = 1:**

  weighted sum = 0 + 1 = 1

  Since 1 > 0.5, the output is 1.

- **When x1 = 1 and x2 = 0:**

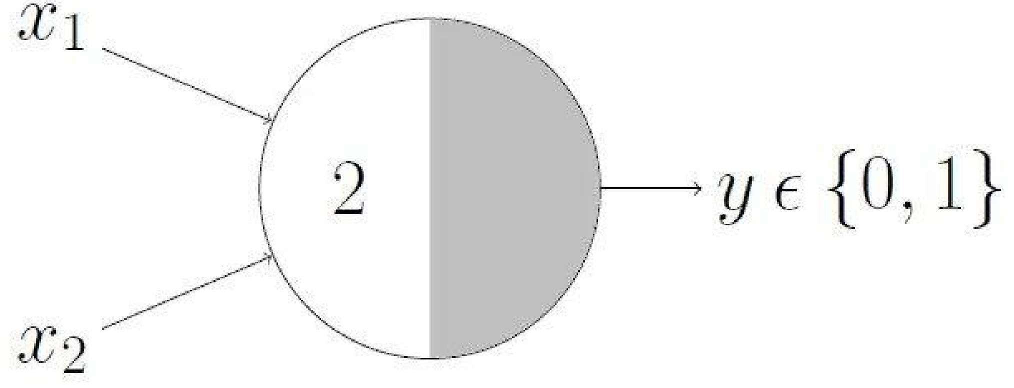
  weighted sum = 1 + 0 = 1

  Since 1 > 0.5, the output is 1.

- **When x1 = 1 and x2 = 1:**

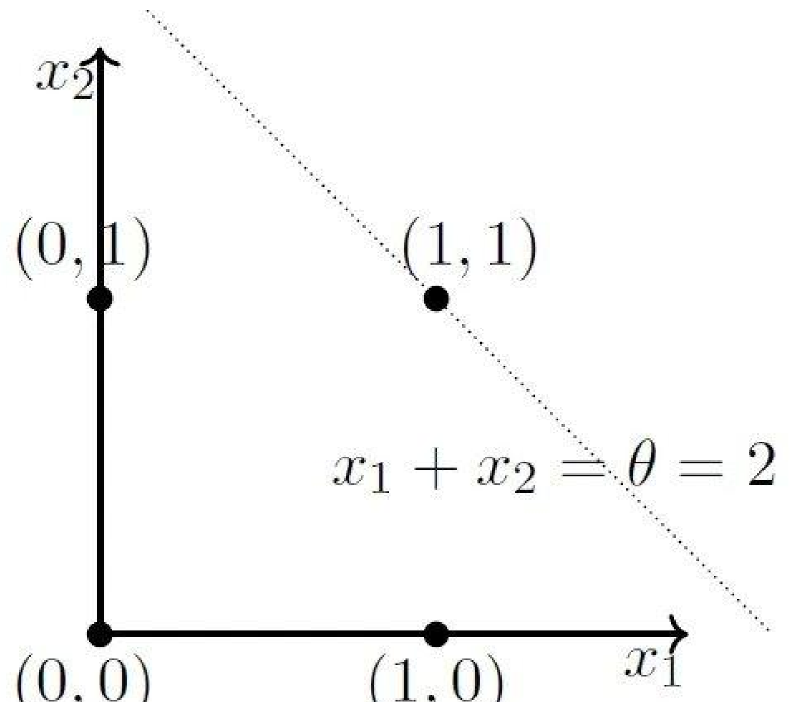
  weighted sum = 1 + 1 = 2

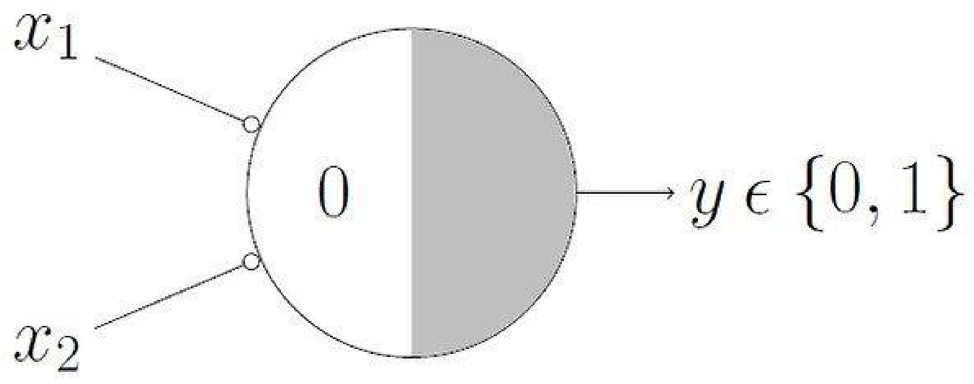
  Since 2 > 0.5, the output is 1.



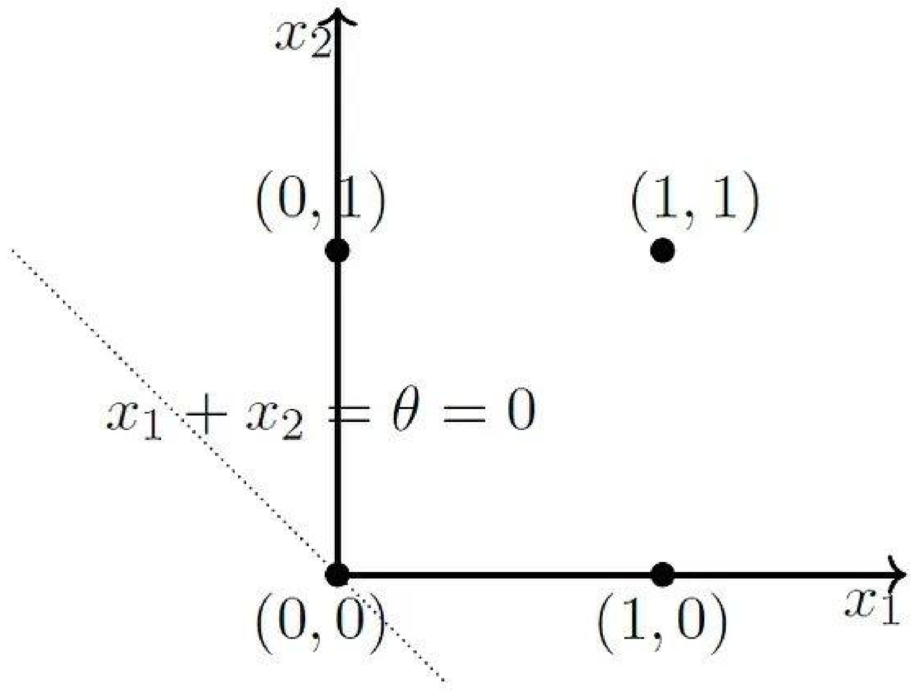
$AND\ function$

$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 2$$





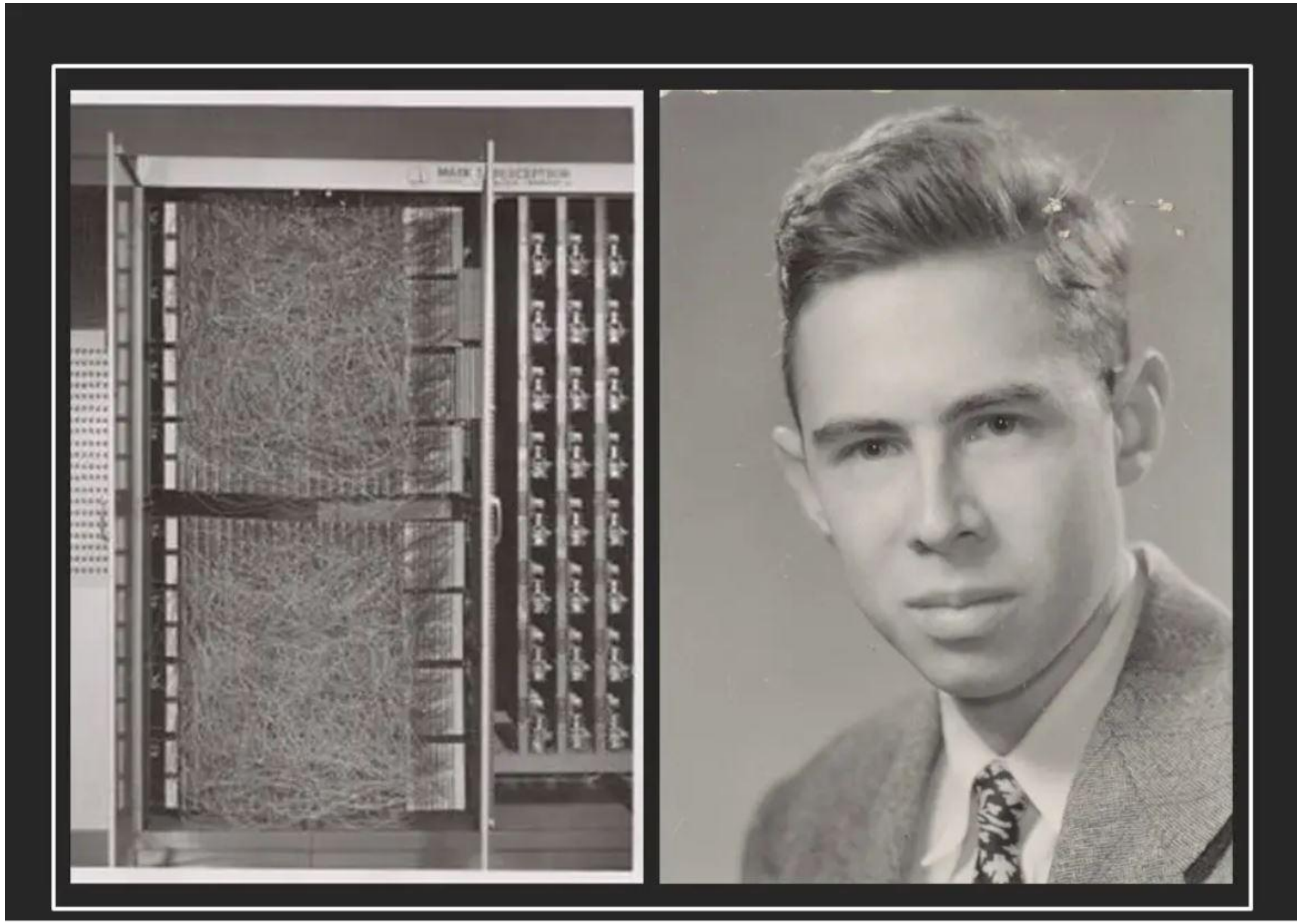
$Tautology\,(always\,ON)$



Limitations

- A single MCP neuron cannot represent the XOR Boolean function or any other nonlinear function.
- All of the synaptic weights are set to unity, implying that all the inputs contribute equally to the output.
- The function considered needs to be hard-coded by the user. It cannot be learned from data.

Rosenblatt



Rosenblatt neuron

- A Rosenblatt neuron is a type of artificial neuron that was first proposed by Frank Rosenblatt in 1957.
- Rosenblatt neurons are inspired by biological neurons, and they are used in artificial neural networks.
- A Rosenblatt neuron has three main components:
  - **Inputs:** The inputs are the signals that are received by the neuron.
  - **Weights:** The weights are the factors that determine how much each input contributes to the output of the neuron.
  - **Threshold:** The threshold is the value that the weighted sum of the inputs must exceed in order for the neuron to fire.

Rosenblatt neuron

- **Inputs:** The neuron receives input signals, represented as real numbers. Each input is associated with a weight that determines its importance or contribution to the neuron's output.
- **Weights:** Each input signal is multiplied by a weight, which represents the strength or significance of that input. The weights can be adjusted during the learning process to optimize the neuron's performance.
- **Weighted Sum:** The neuron computes the weighted sum of the inputs by multiplying each input with its corresponding weight and summing them up.
- **Activation Function:** The weighted sum is passed through an activation function that determines the output of the neuron. The most common activation function used in the Rosenblatt neuron is a ***step function or threshold function.*** If the weighted sum exceeds a certain threshold, the neuron outputs a positive value (e.g., 1); otherwise, it outputs a negative value (e.g., 0).

Rosenblatt neuron

- **Learning Algorithm:** The Rosenblatt neuron employs a learning algorithm known as the Perceptron learning rule. ***This rule adjusts the weights of the inputs based on the error between the actual output and the desired output.*** The aim is to iteratively update the weights to improve the neuron's ability to correctly classify or discriminate between different input patterns.
- The Rosenblatt neuron is particularly **suited for binary classification tasks**, where it learns to separate input patterns into two distinct classes based on a decision boundary defined by the weights and the activation function.
- While the Rosenblatt neuron is a simple model, it played a significant role in the development of neural networks and machine learning.
- It demonstrated the concept of learning from examples and the ability to solve linearly separable classification problems.
- Although limited to linearly separable problems, the Rosenblatt neuron paved the way for more complex neural network architectures and learning algorithms.

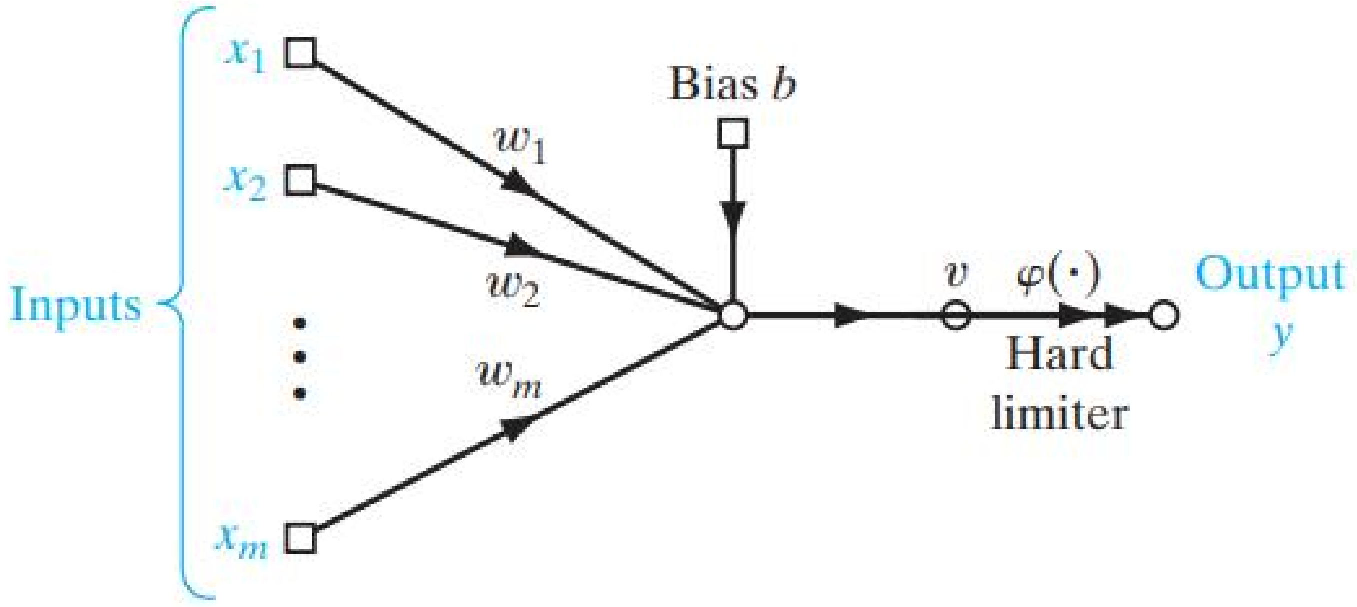
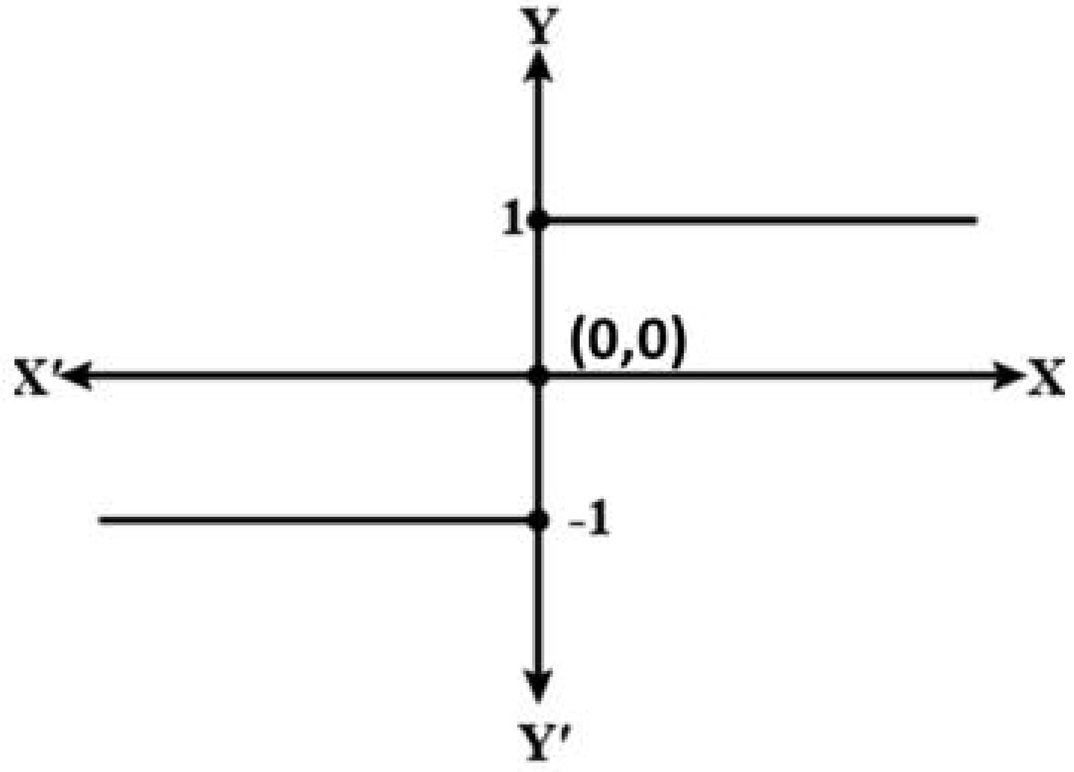


FIGURE 1.1 Signal-flow graph of the perceptron.

$$Sgn(x) = \begin{cases} -1\ ,\ if\ x < 0 \\ 0,\ if, x = 0 \\ 1,\ if, x > 0 \end{cases}$$

Graph of Signum Function



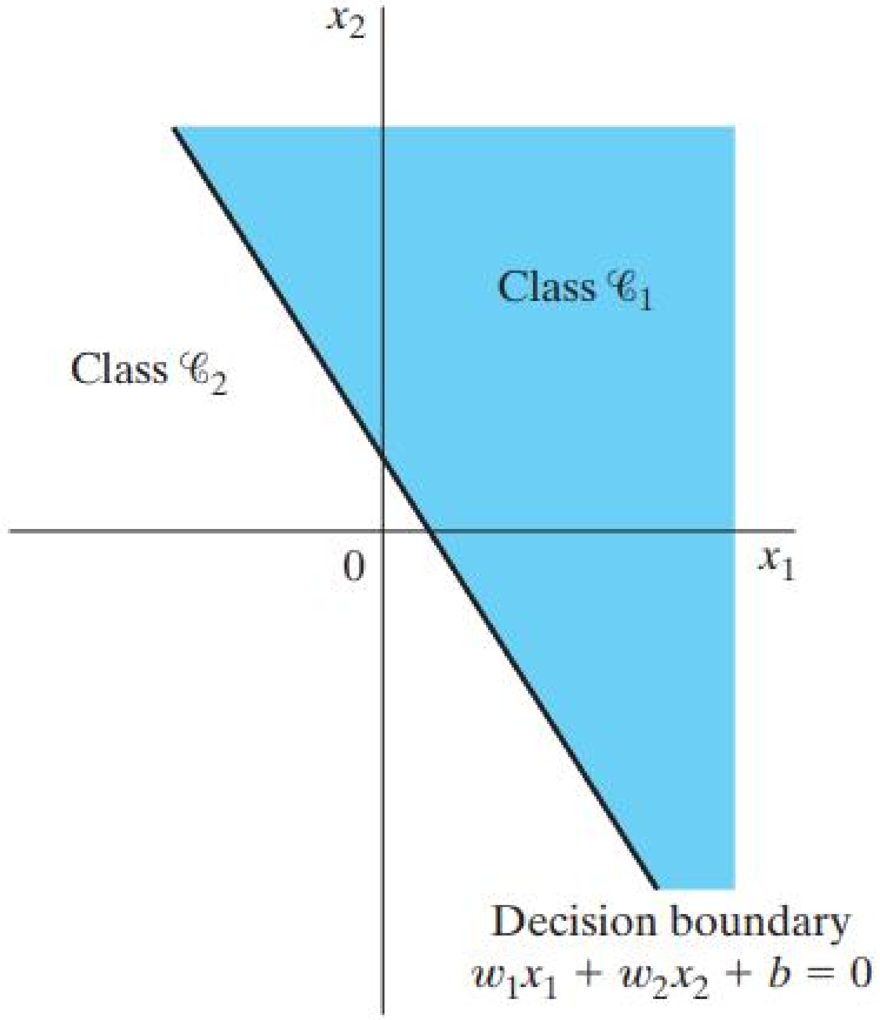
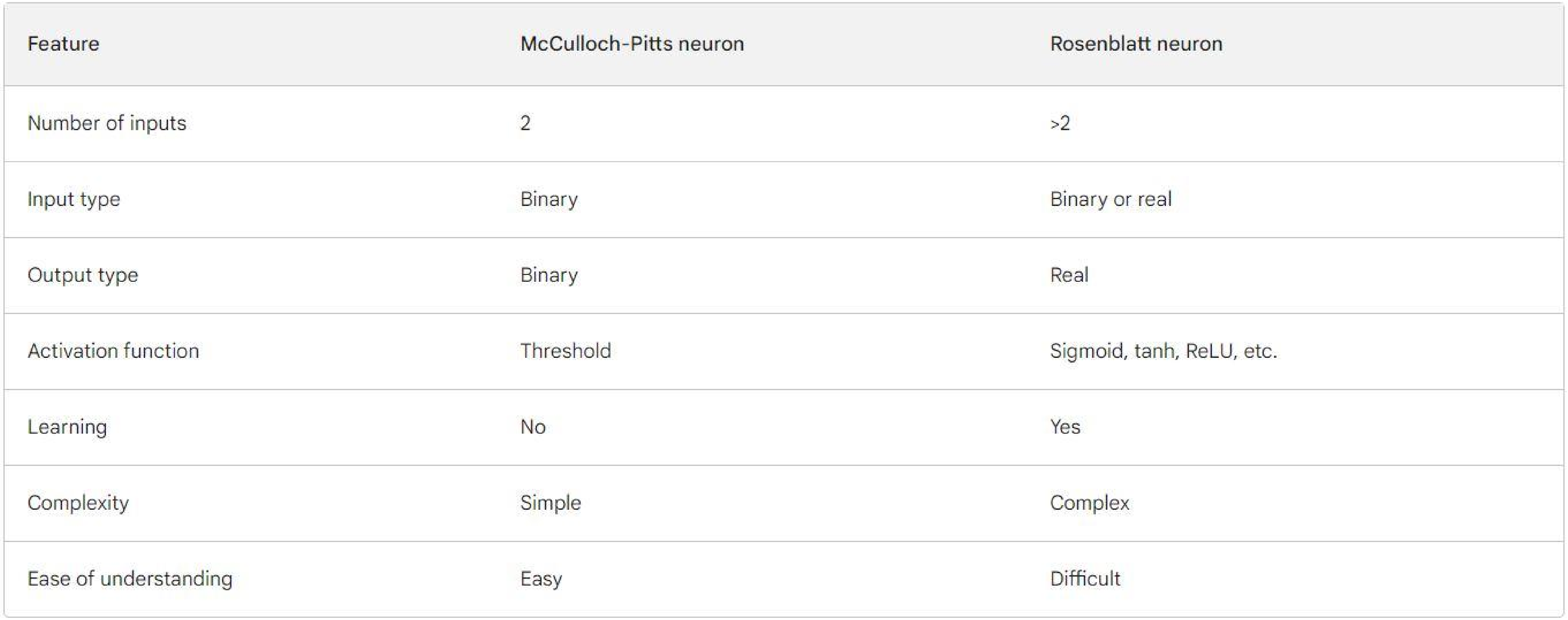


FIGURE 1.2 Illustration of the hyperplane (in this example, a straight line) as decision boundary for a two-dimensional, two-class pattern-classification problem.



Feature	McCulloch-Pitts neuron	Rosenblatt neuron
Number of inputs	2	>2
Input type	Binary	Binary or real
Output type	Binary	Real
Activation function	Threshold	Sigmoid, tanh, ReLU, etc.
Learning	No	Yes
Complexity	Simple	Complex
Ease of understanding	Easy	Difficult

Perceptron

A perceptron is the simplest form of a neural network, consisting of a single layer of artificial neurons (also known as perceptrons) with no hidden layers.

Each perceptron takes a set of input values, applies weights to them, and passes the weighted sum through an activation function to produce an output.

The output can be binary (0 or 1) or continuous. Perceptrons are mainly used for binary classification problems, where they learn to separate input data into two classes based on a linear decision boundary.

Multilayer Neural Networks

A multilayer neural network, also known as a feedforward neural network or a deep neural network, consists of multiple layers of artificial neurons arranged in a sequential manner.

It includes an input layer, one or more hidden layers, and an output layer.

The neurons in each layer are connected to the neurons in the subsequent layer, forming a network of interconnected nodes.

The information flows in one direction, from the input layer through the hidden layers to the output layer.

Multilayer Neural Networks

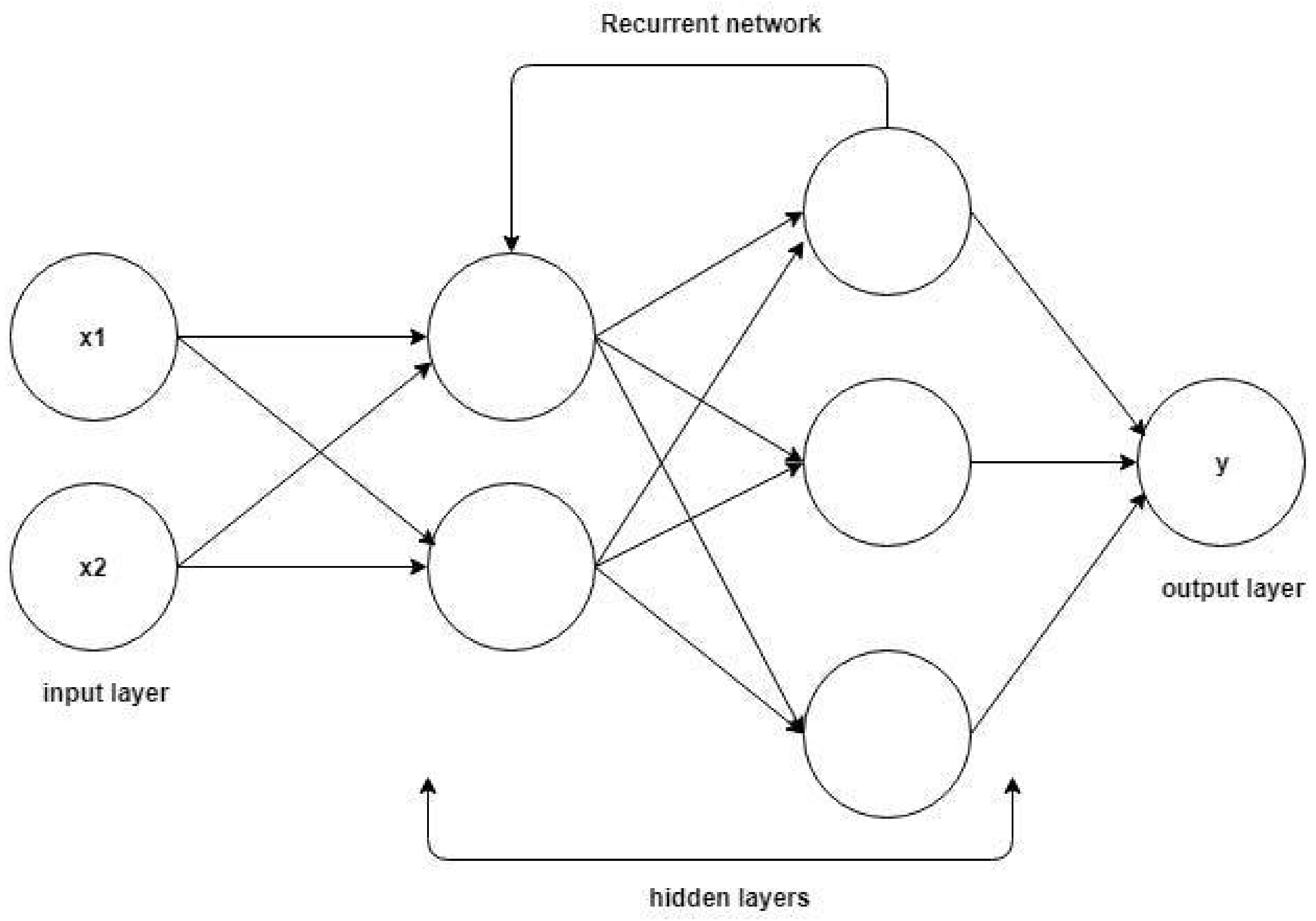
In a multilayer neural network, each neuron receives inputs from the neurons in the previous layer, applies weights to them, and passes the weighted sum through an activation function.

The activation function introduces non-linearity, allowing the network to learn complex patterns and relationships in the data.

The weights in the network are adjusted using backpropagation, where the error is propagated backward through the network, enabling the network to update its weights and improve its performance.

Recursive Neural Networks

- Recursive neural networks (RNNs) are a type of neural network that can process structured data, such as hierarchical data or data with recursive relationships.
- They are designed to handle inputs that have a recursive or tree-like structure, where the relationships between the elements are important.
- In an RNN, instead of processing the data sequentially layer by layer, the network recursively applies the same set of weights and biases to nodes in the input structure.
- This recursive application allows the network to capture the compositional nature of the data, where the meaning of a structure is built from smaller structures.
- Recursive neural networks are commonly used in natural language processing tasks, such as sentiment analysis, parsing, and machine translation, where the syntactic and semantic structures of sentences are crucial for understanding the meaning.


Recurrent network
x1
x2
y
input layer
output layer
hidden layers

Gradient Descent

- Gradient Descent is known as one of the most commonly used optimization algorithms to train machine learning models by means of minimizing errors between actual and expected results.
- Further, gradient descent is also used to train Neural Networks.
- The main objective of gradient descent is to minimize the convex function using iteration of parameter updates.
- Once these machine learning models are optimized, these models can be used as powerful tools for Artificial Intelligence and various computer science applications.

Gradient Descent

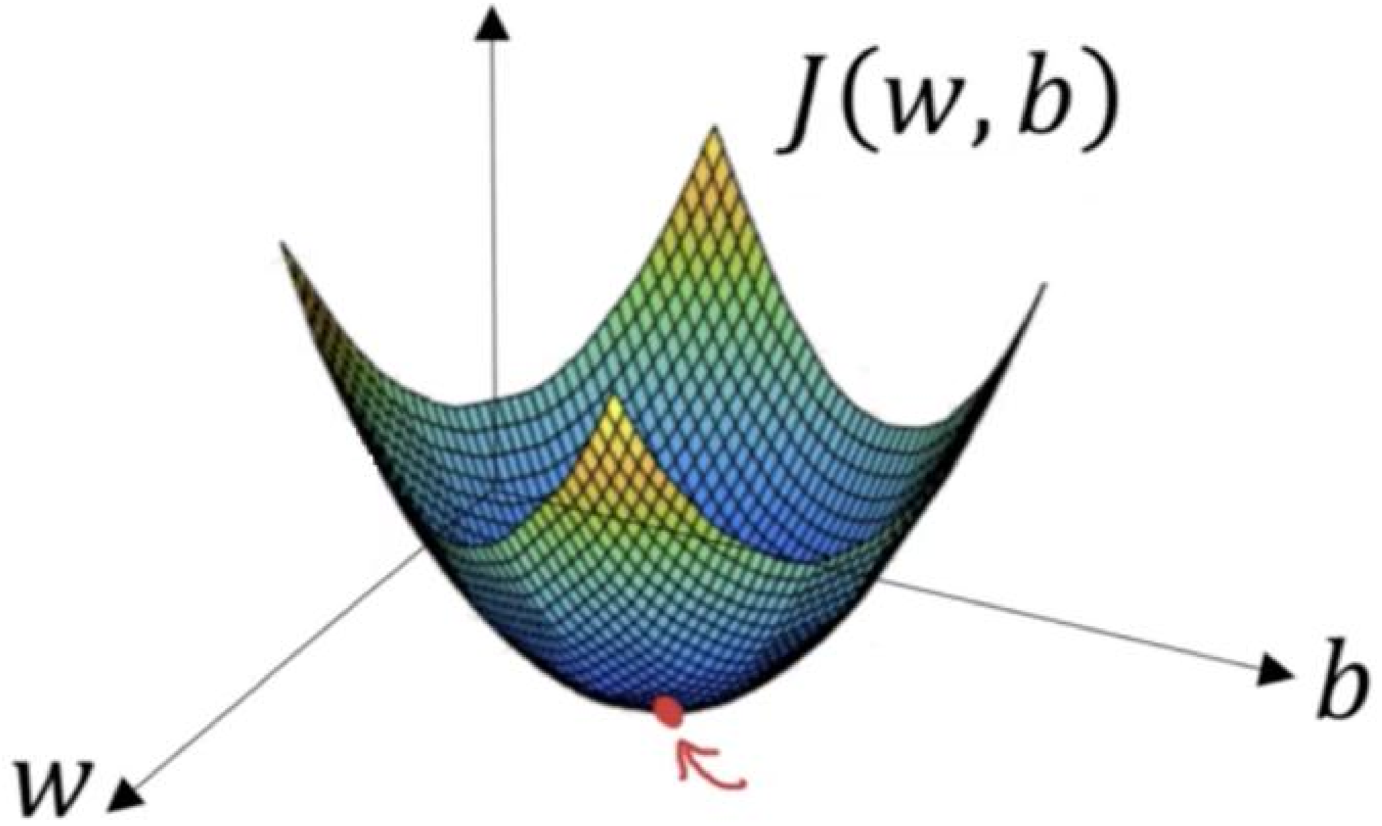
- Gradient Descent is defined as one of the most commonly used iterative optimization algorithms of machine learning to train the machine learning and deep learning models.
- It helps in finding the local minimum of a function.
- If we move towards a negative gradient or away from the gradient of the function at the current point, it will give the local minimum of that function.
- Whenever we move towards a positive gradient or towards the gradient of the function at the current point, we will get the local maximum of that function.

Gradient Descent

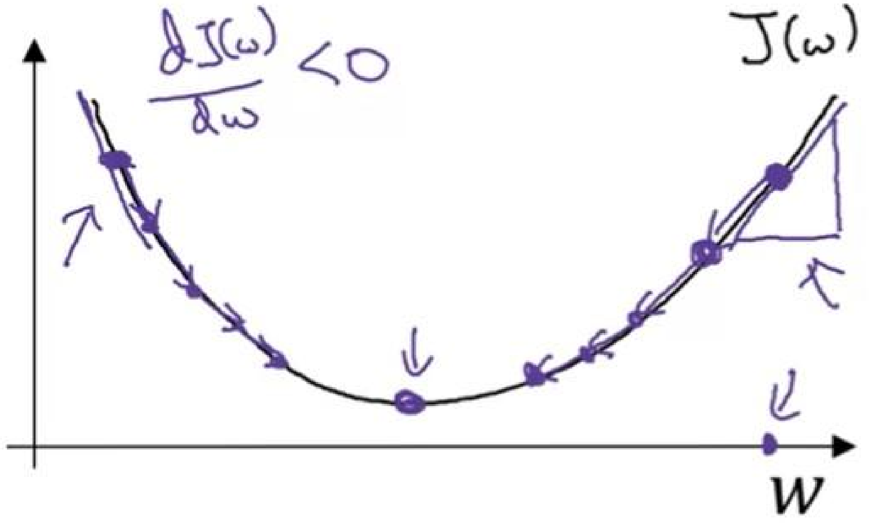
Recap: $\hat{y} = \sigma(w^T x + b), \ \sigma(z) = \frac{1}{1+e^{-z}}$

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

Want to find w, b that minimize J(w, b)



Gradient Descent



$$\text{Repeat } \{$$

$$\left(w := w - \overset{\text{learning rate}}{\alpha}\ \underbrace{\boxed{\frac{dJ(w)}{dw}}}_{\text{"}dw\text{"}} \right.$$

$$\uparrow \quad \uparrow$$

$$\}$$

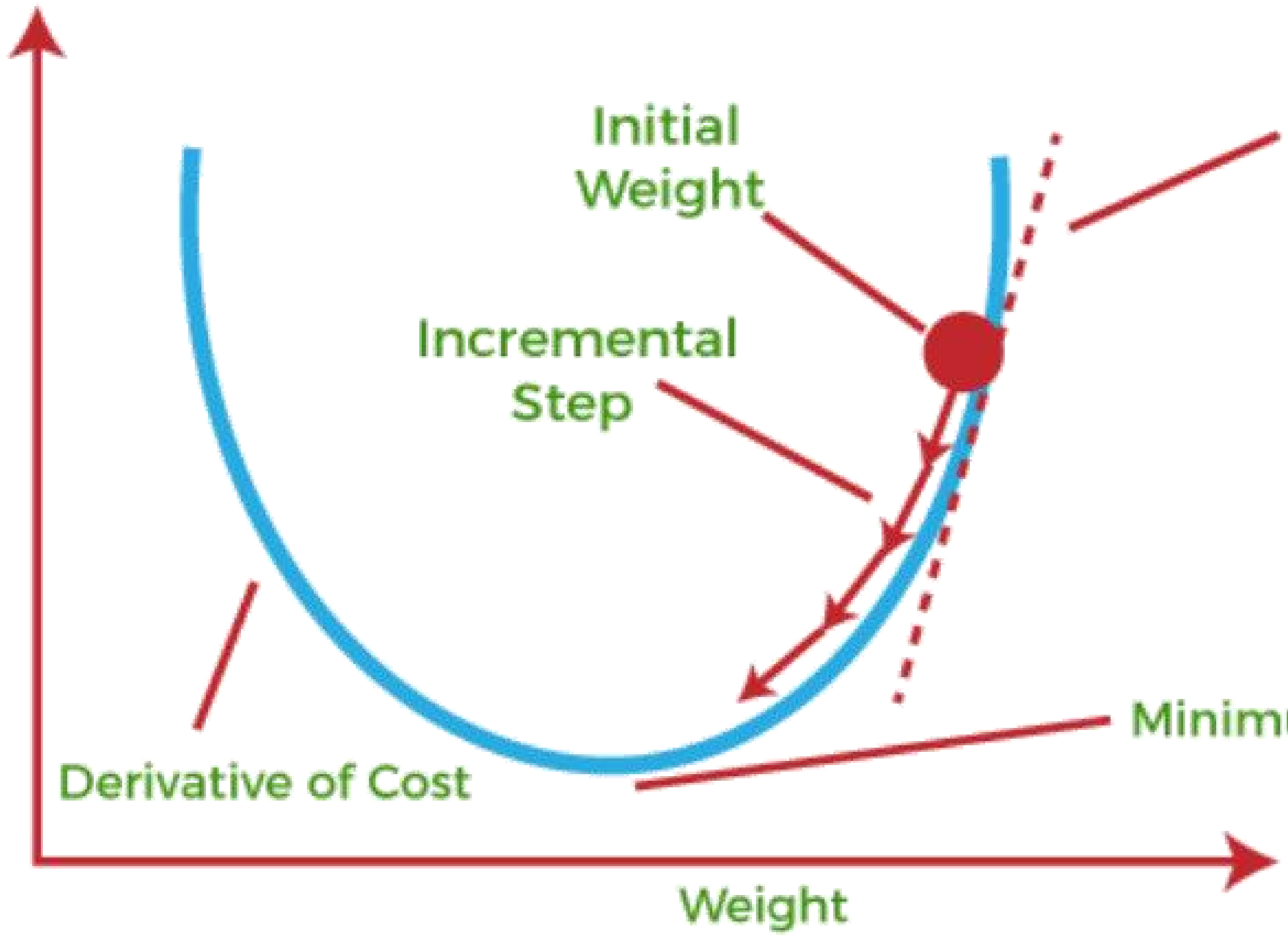
$$w := w - \alpha \underline{dw}$$

$$\underbrace{\frac{dJ(w)}{dw}} = ?$$

$$J(w, b)$$

$$w := w - \alpha \frac{dJ(w,b)}{dw}$$

$$b := b - \alpha \frac{dJ(w,b)}{db}$$


Initial Weight
Gradient
Incremental Step
Minimum Cost
Derivative of Cost
Weight

Gradient Descent

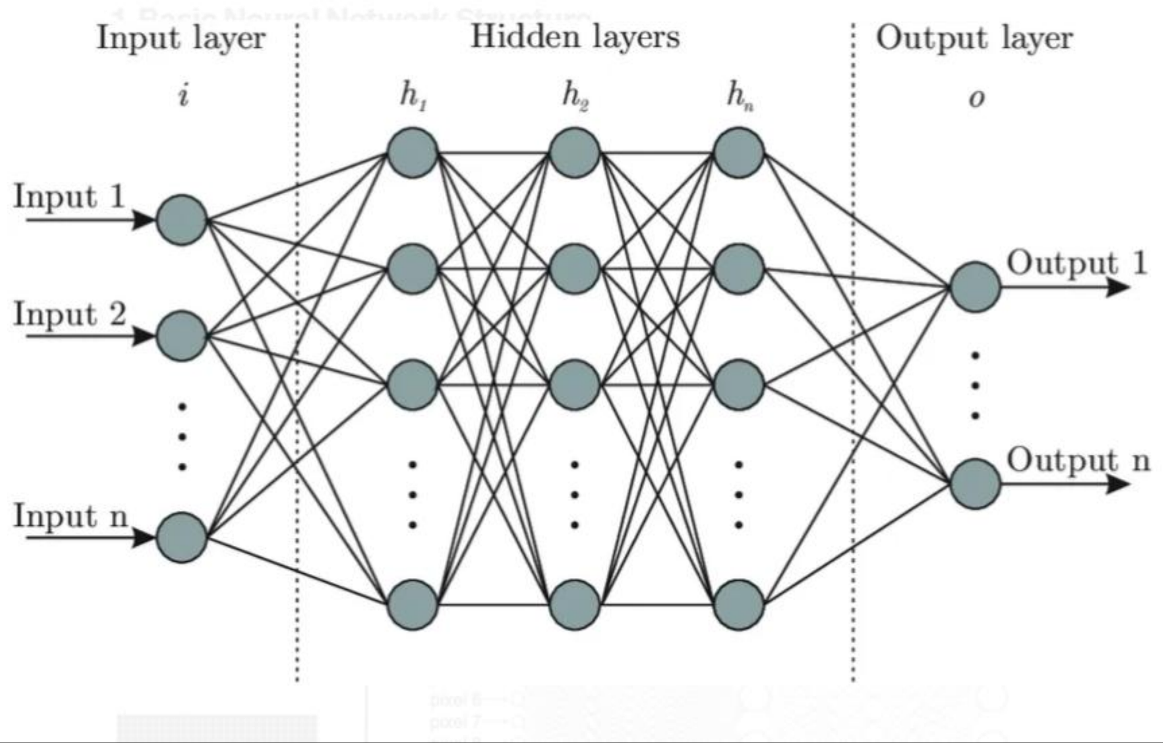
- This entire procedure is known as Gradient Ascent, which is also known as steepest descent.
- The main objective of using a gradient descent algorithm is to minimize the cost function using iteration.
- To achieve this goal, it performs two steps iteratively:
  - Calculates the first-order derivative of the function to compute the gradient or slope of that function.
  - Move away from the direction of the gradient, which means slope increased from the current point by alpha times, where Alpha is defined as Learning Rate. It is a tuning parameter in the optimization process which helps to decide the length of the steps.

Backpropagation

- The backpropagation algorithm is a fundamental method used for training artificial neural networks.
- It is an abbreviation for "backward propagation of errors," which describes the way errors are propagated backward through the network to update the weights.
- Backpropagation aims to minimize the error between the actual output of the neural network and the desired output.

Neural Network Structure

- **Input Layer:** The initial layer that receives input data.
- **Hidden Layers:** Intermediate layers that process inputs from the previous layer with weights and biases.
- **Output Layer:** The final layer that provides the output of the network.



Forward Pass

- Input data is passed through the network layer by layer.
- Each neuron computes a weighted sum of its inputs, applies an activation function, and passes the result to the next layer.

$$\boldsymbol{Y = f\left(\sum w_i x_i + b \right)}$$

where y is the output, f is the activation function, w_i are weights, x_i are inputs, and b is the bias.

- **Loss Function:**
  - A function that measures the difference between the actual output and the desired output (e.g., Mean Squared Error for regression, Cross-Entropy Loss for classification).

Backward Pass (Backpropagation)

- The error is calculated at the output layer using the loss function.
- This error is then propagated backward through the network.
- Gradients (partial derivatives of the loss function with respect to each weight) are calculated for each weight.

Backward Pass (Backpropagation)

- Weights are adjusted using gradient descent or one of its variants.
- The weights are updated in the direction that reduces the error (gradient descent):

$$\boldsymbol{w_{ij} \leftarrow w_{ij} - \eta \ (\partial E / \partial w_{ij})}$$

- where w_{ij} is the weight between neurons i and j, E is the error, and η is the learning rate.

In Summary:

- **Initialization:** Initialize weights and biases, often with small random values.
- **Forward Propagation:** Compute the outputs for each layer from the input to the output layer.
- **Compute Error:** Calculate the error at the output layer using the loss function.
- **Backward Propagation:** Compute the gradient of the loss function with respect to each weight using the chain rule of calculus. Propagate the error backward from the output layer to the hidden layers.
- **Update Weights:** Adjust weights and biases using the computed gradients and the learning rate.
- **Repeat:** Repeat the process for a set number of iterations or until the error is minimized to an acceptable level.

Common Activation Functions:

- **Sigmoid**:

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

- **Tanh**:

$$f(x) = \tanh(x)$$

- **ReLU (Rectified Linear Unit)**:

$$f(x) = \max(0, x)$$