DRONACHARYA
College of Engineering

Section D

Machine Learning

Outline

- Artificial intelligence in 21st century
- Learning
- Machine learning
- Supervised learning
- How brain works
- Neural network and Artificial neural networks
- Simple neuron - Perceptron

Artificial Intelligence

- The capacity of a computer to perform operations analogous to learning and decision making in humans, as by an expert system, a program for CAD or CAM, or a program for the perception and recognition of shapes in computer vision systems

Business Intelligence

- The process of gathering information about a business or industry matter; a broad range of applications and technologies for gathering, storing, analyzing, and providing access to data to help make business decisions
- BI

Computational Intelligence

- An offshoot of artificial intelligence. As an alternative to GOFAI (Good Old-Fashioned Artificial Intelligence) it rather relies on heuristic algorithms such as in fuzzy systems, neural networks and evolutionary computation. In addition, computational intelligence also embraces techniques that use Swarm intelligence, Fractals and Chaos Theory, Artificial immune systems, Wavelets, etc

Traditional Artificial Intelligence…

- Success
  - Chess playing
  - Mathematical theorem prove
  - Expert systems
- Failure
  - Face Identification
  - Natural Language processing
  - Robotics

Symbolic Artificial Intelligence

- **Symbol processing**
- **Logic programming**
- **List processing**
- **Top down**
- **High level processing**

Distributed Artificial Intelligence

- **Grounded in experience**
- **Information held in a distributed manner**
- **Information held locally**
- **Bottom up**
- **No overall control**
- **Learn from experience**
- **Based on Biological Models**
    - Artificial Neural Networks
    - Genetic Algorithms
    - Artificial Life

Artificial Neural Networks

- Forecast time series
- Control robots
- Pattern recognition
- Noise removal
- Digit recognition
- Personal identification
- Optimise portfolios
- Data mining

Genetic Algorithms

- Based on evolution
- Survival of the fittest
- The survivors get more chances to breed
- The species becomes fitter generation by generation … but so do the enemies
- Change is inherent in the process
- Applications:
  - Optimisation in general
  - Traveling Salesman Problem
  - Timetables
  - Best shares portfolio

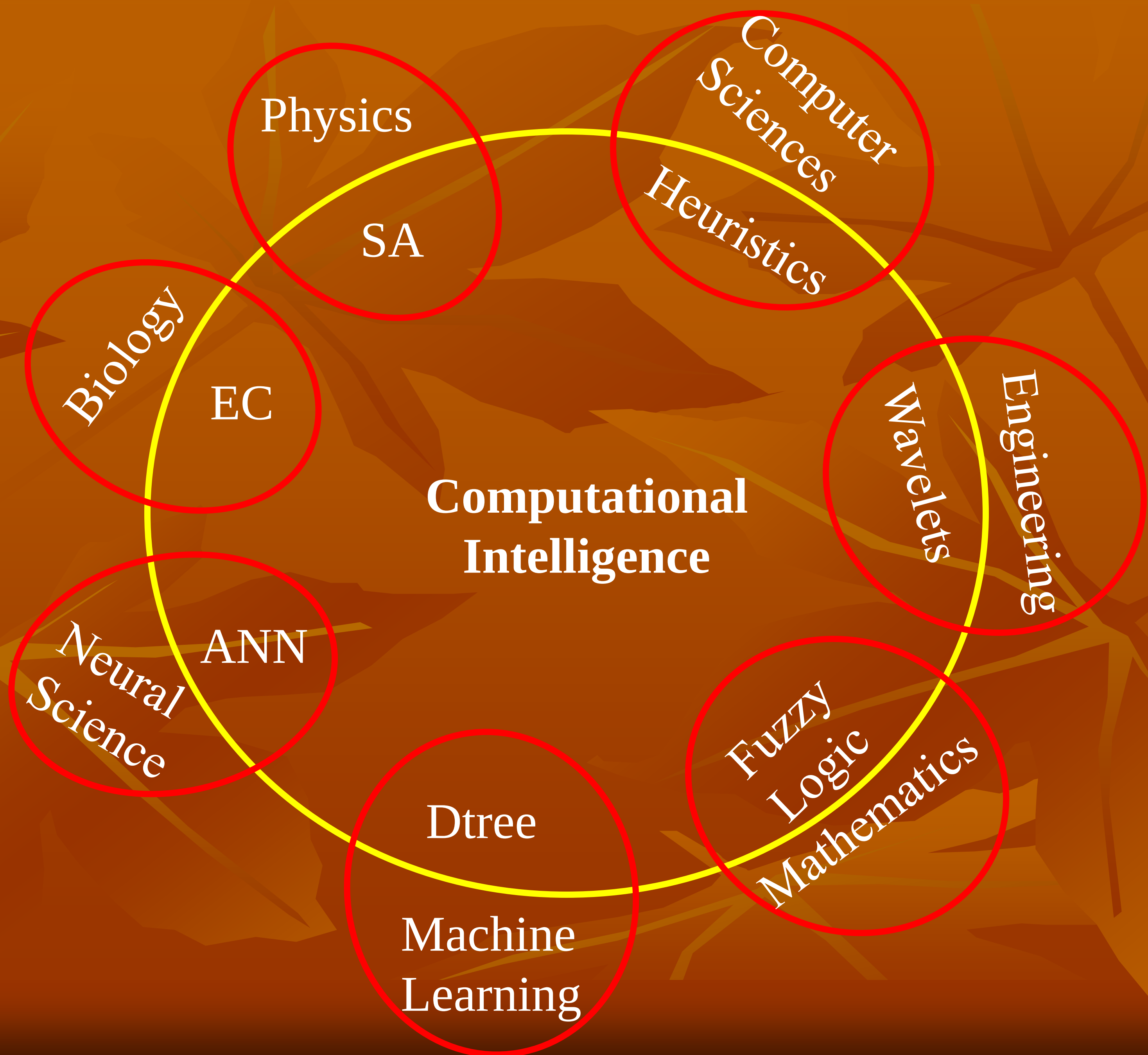
Artificial Life

- Mixture of evolution and learning
- Evolution of Language
- Evolution of Cooperation

Computational Intelligence

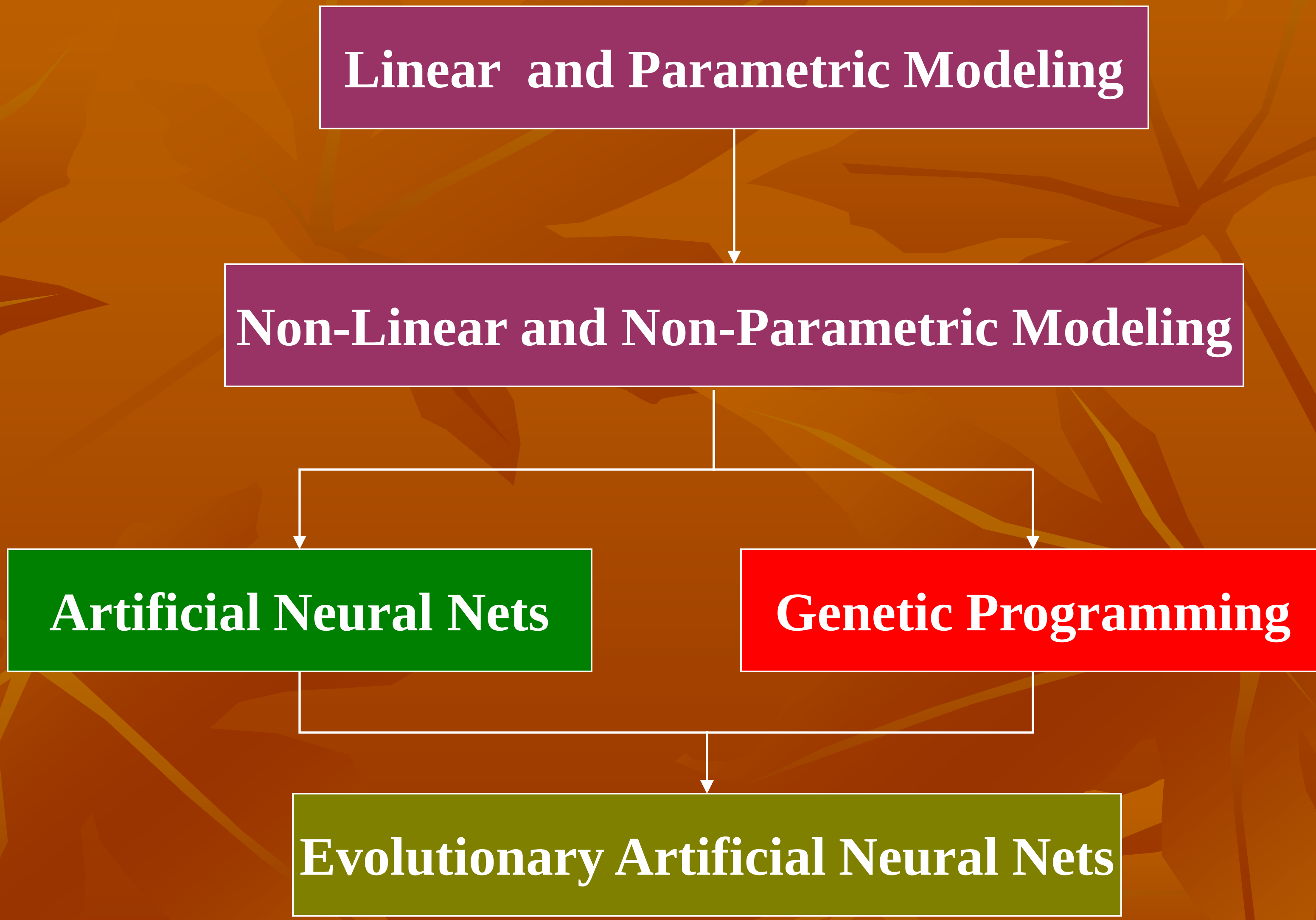
- Defining "Computational Intelligence" is not straightforward. Several expressions compete to name the same interdisciplinary area.
- It is difficult, if not impossible, to accommodate in a formal definition disparate areas with their own established individualities such as fuzzy sets, neural networks, evolutionary computation, machine learning, Bayesian reasoning, etc.

- "Computational Intelligence" is rather the intuition behind the synergism between these and many more, at the verge of Computer Sciences, Mathematics and Engineering. Bringing together diverse expertise and experience can enrich each of the participating discipline and foster new research perspectives in the broad field of Computational Intelligence.

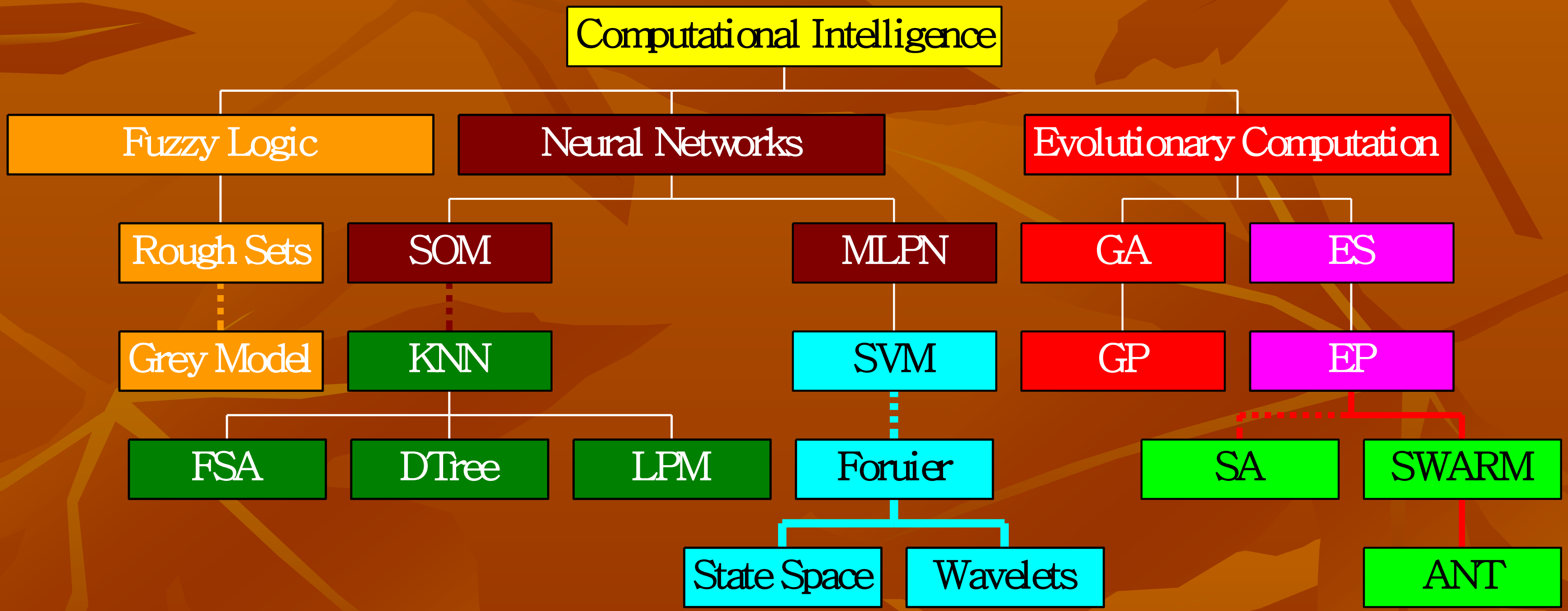

Physics
SA
Computer
Sciences
Heuristics
Biology
EC
Computational
Intelligence
Wavelets
Engineering
Neural
Science
ANN
Fuzzy
Logic
Mathematics
Dtree
Machine
Learning

Soft Computing

- **According to Prof. Zadeh:**
- "...in contrast to traditional hard computing, soft computing exploits the tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, low solution-cost, and better rapport with reality”


Linear and Parametric Modeling
Non-Linear and Non-Parametric Modeling
Artificial Neural Nets
Genetic Programming
Evolutionary Artificial Neural Nets

The Family Tree



Learning

- Learning is a fundamental and essential characteristic of biological neural networks.
- The ease with which they can learn led to attempts to emulate a biological neural network in a computer.

3 main types of learning

- **Supervised learning**
  - learning with a teacher
- **Unsupervised learning**
  - Learning from pattern
- **Reinforcement learning**
  - Learning through experiences

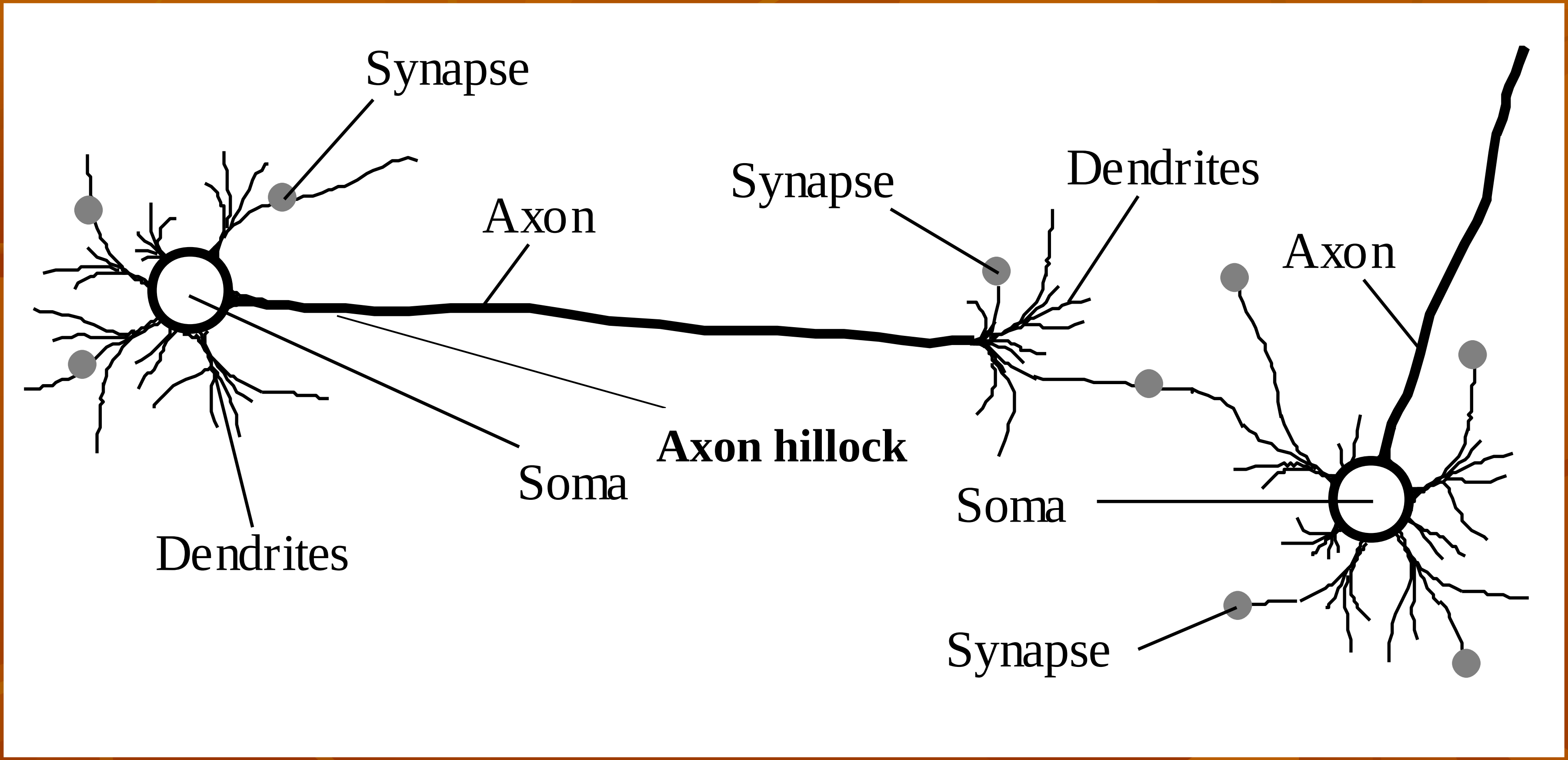
Machine Learning

- Machine learning involves adaptive mechanisms that enable computers to learn from experience, learn by example and learn by analogy. Learning capabilities can improve the performance of an intelligent system over time.
- The most popular approaches to machine learning are **artificial neural networks** and **genetic algorithms**.

How the brain works

- A **neural network** can be defined as a model of reasoning based on the human brain. The brain consists of a densely interconnected set of nerve cells, or basic information-processing units, called **neurons**.
- The human brain incorporates nearly 10 billion neurons and 60 trillion connections, *synapses*, between them. By using multiple neurons simultaneously, the brain can perform its functions much faster than the fastest computers in existence today.
- Each neuron has a very simple structure, but an army of such elements constitutes a tremendous processing power.
- A neuron consists of a cell body, **soma**, a number of fibers called **dendrites**, and a single long fiber called the **axon**.

Biological neural network

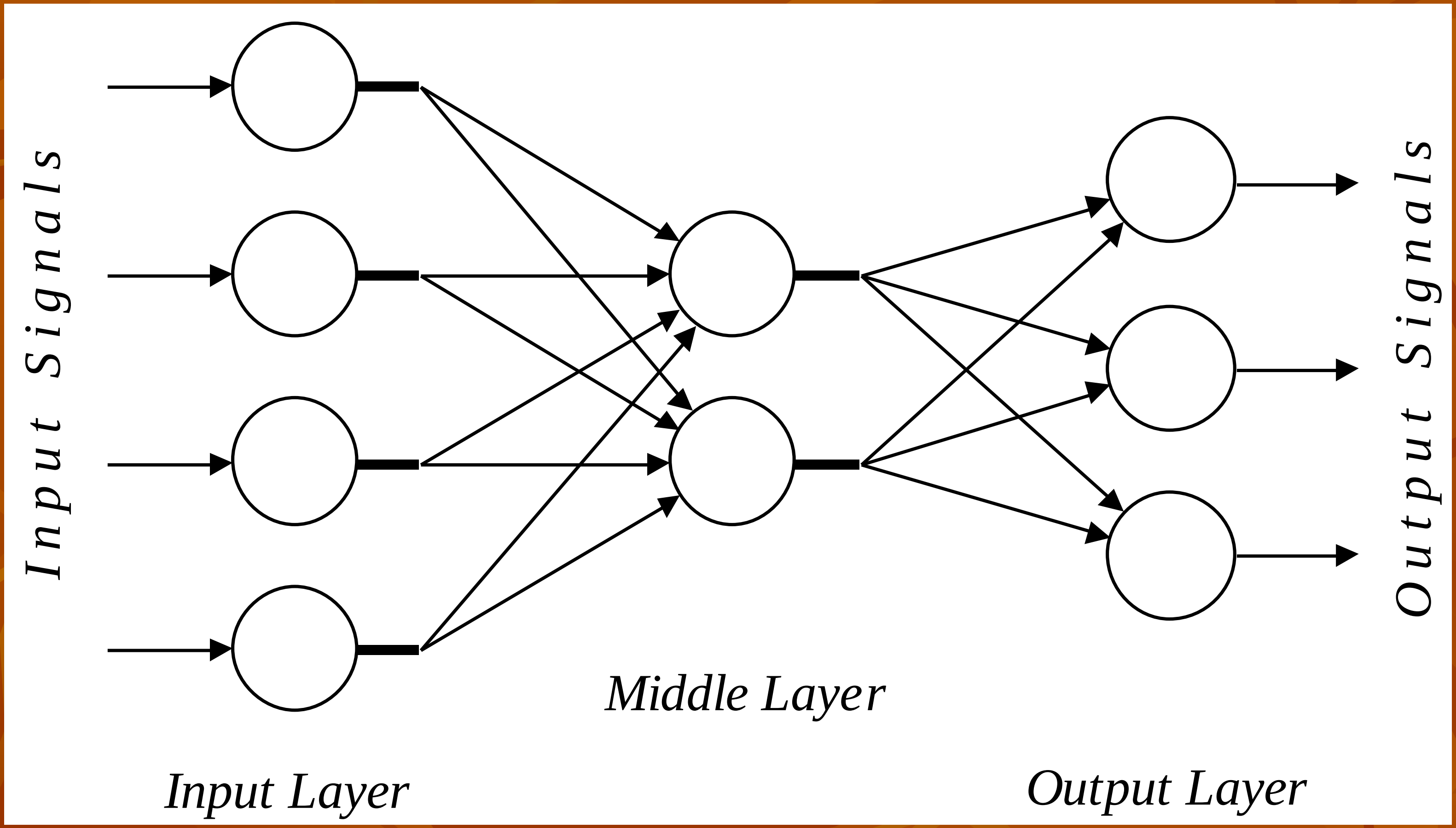


- Our brain can be considered as a highly complex, non-linear and parallel information-processing system.
- Information is stored and processed in a neural network simultaneously throughout the whole network, rather than at specific locations. In other words, in neural networks, both data and its processing are **global** rather than local.

Artificial Neural Networks

- An artificial neural network consists of a number of very simple processors, also called **neurons**, which are analogous to the biological neurons in the brain.
- The neurons are connected by weighted links passing signals from one neuron to another.
- The output signal is transmitted through the neuron's outgoing connection. The outgoing connection splits into a number of branches that transmit the same signal. The outgoing branches terminate at the incoming connections of other neurons in the network.

Architecture of an ANN

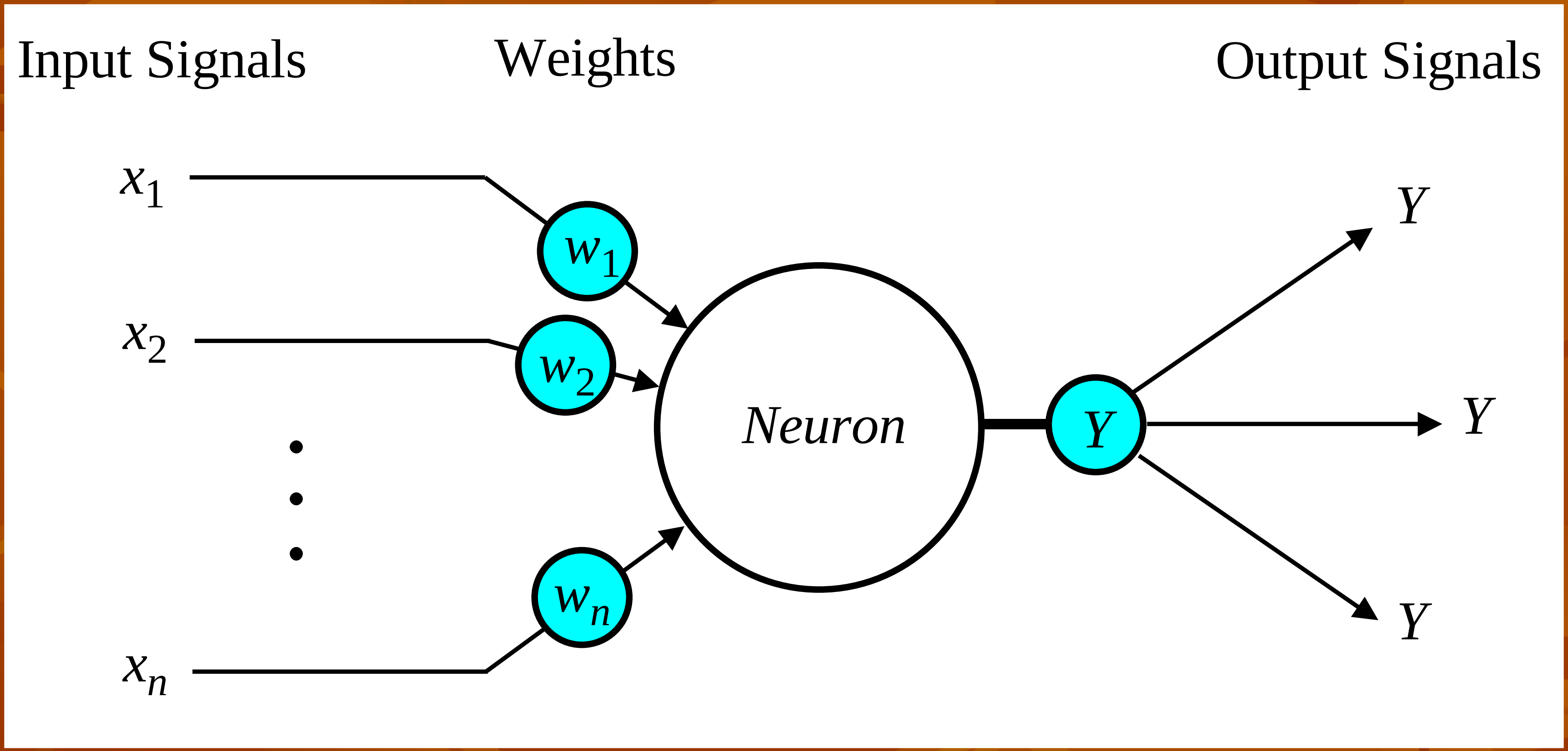


Analogy between biological and artificial neural networks

Biological Neural Network	*Artificial Neural Network*
Soma	Neuron
Dendrite	Input
Axon	Output
Synapse	Weight

The neuron as a simple computing element

Diagram of a neuron



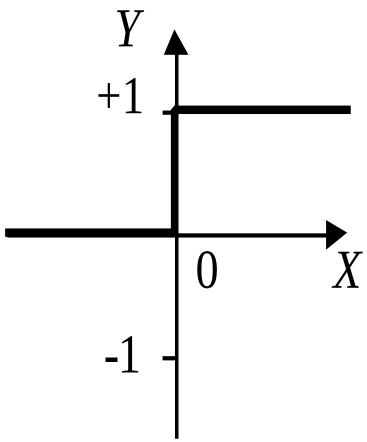
- The neuron computes the weighted sum of the input signals and compares the result with a **threshold value**, θ. If the net input is less than the threshold, the neuron output is –1. But if the net input is greater than or equal to the threshold, the neuron becomes activated and its output attains a value +1.
- The neuron uses the following transfer or **activation function**:

$$X = \sum_{i=1}^{n} x_i w_i \qquad\qquad Y = \begin{cases} +1, \text{ if } X \geq \theta \\ -1, \text{ if } X < \theta \end{cases}$$

- This type of activation function is called a **sign function**.

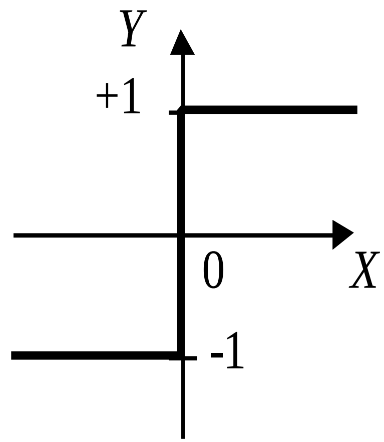
Activation functions of a neuron

Step function



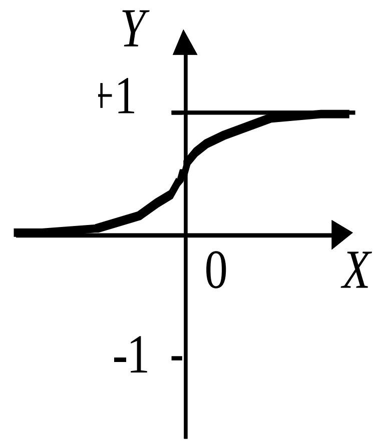
$$Y^{step} = \begin{cases} 1, & \text{if } X \ge 0 \\ 0, & \text{if } X < 0 \end{cases}$$

Sign function



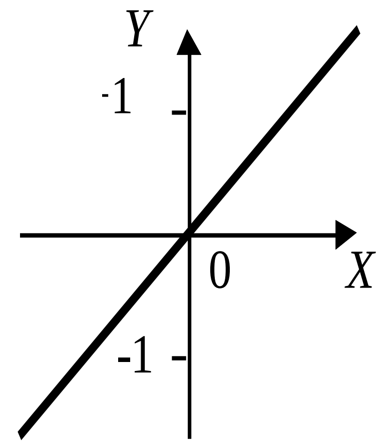
$$Y^{sign} = \begin{cases} +1, & \text{if } X \ge 0 \\ -1, & \text{if } X < 0 \end{cases}$$

Sigmoid function



$$Y^{sigmoid} = \frac{1}{1+e^{-X}}$$

Linear function

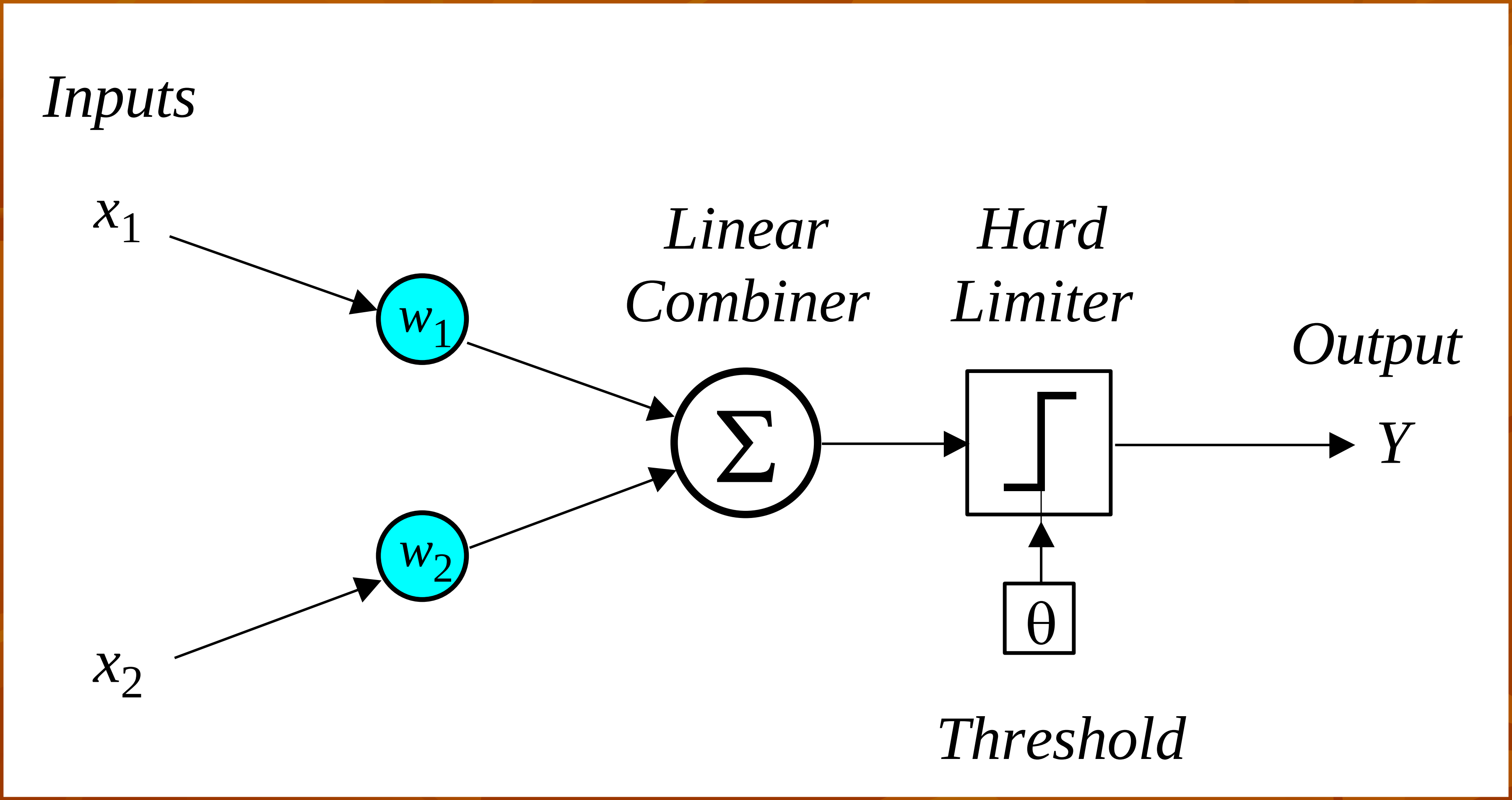


$$Y^{linear} = X$$

Perceptron

- In 1958, **Frank Rosenblatt** introduced a training algorithm that provided the first procedure for training a simple ANN: a **perceptron**.
- The perceptron is the simplest form of a neural network. It consists of a single neuron with *adjustable* synaptic weights and a *hard limiter*.
- The operation of Rosenblatt's perceptron is based on the **McCulloch and Pitts neuron model**. The model consists of a linear combiner followed by a hard limiter.
- The weighted sum of the inputs is applied to the hard limiter, which produces an output equal to +1 if its input is positive and –1 if it is negative.

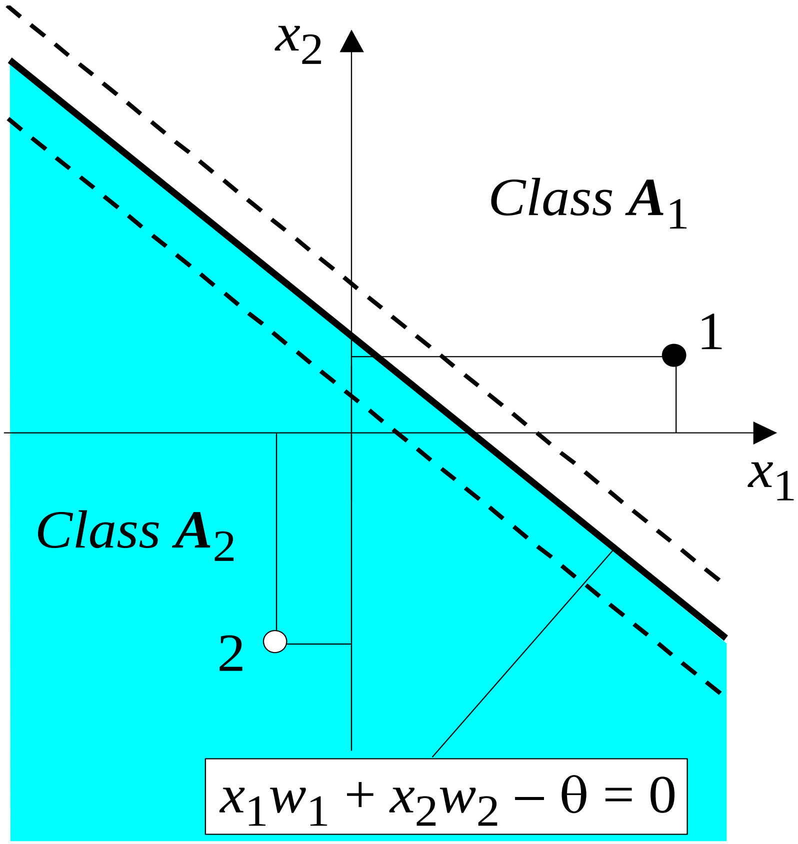
Single-layer two-input perceptron



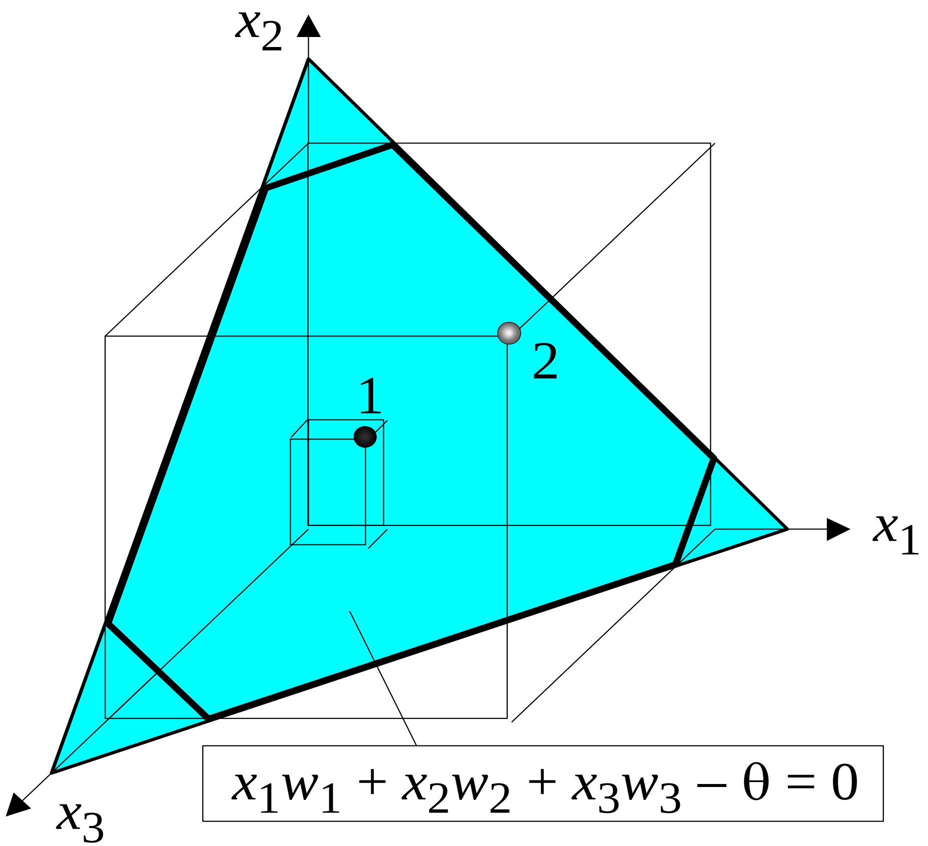
- The aim of the perceptron is to classify inputs, $x_1, x_2, \ldots, x_n$, into one of two classes, say $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$.
- In the case of an elementary perceptron, the n-dimensional space is divided by a *hyperplane* into two decision regions. The hyperplane is defined by the ***linearly separable*** **function**:

$$\sum_{i=1}^{n} x_i w_i - \theta = 0$$

Linear separability in the perceptrons



(*a*) Two-input perceptron.



(*b*) Three-input perceptron.

How does the perceptron learn its classification tasks?

This is done by making small adjustments in the weights to reduce the difference between the actual and desired outputs of the perceptron. The initial weights are randomly assigned, usually in the range [–0.5, 0.5], and then updated to obtain the output consistent with the training examples.

- If at iteration p, the actual output is $Y(p)$ and the desired output is $Y_d(p)$, then the error is given by:

$$e(p) = Y_d(p) - Y(p)$$ where p = 1, 2, 3, . . .

Iteration p here refers to the pth training example presented to the perceptron.

- If the error, $e(p)$, is positive, we need to increase perceptron output $Y(p)$, but if it is negative, we need to decrease $Y(p)$.

The perceptron learning rule

$$w_i(p+1) = w_i(p) + \alpha \cdot x_i(p) \cdot e(p)$$

where p = 1, 2, 3, . . .

α is the **learning rate**, a positive constant less than unity.

The perceptron learning rule was first proposed by **Rosenblatt** in 1960. Using this rule we can derive the perceptron training algorithm for classification tasks.

Perceptron's training algorithm

Step 1: Initialisation

Set initial weights $w_1, w_2, \ldots, w_n$ and threshold θ to random numbers in the range [–0.5, 0.5].

If the error, $e(p)$, is positive, we need to increase perceptron output $Y(p)$, but if it is negative, we need to decrease $Y(p)$.

Perceptron's training algorithm (continued)

Step 2: Activation

Activate the perceptron by applying inputs $x_1(p)$, $x_2(p)$,…, $x_n(p)$ and desired output $Y_d(p)$.
Calculate the actual output at iteration $p = 1$

$$Y(p) = step\left[\sum_{i=1}^{n} x_i(p)\, w_i(p) - \theta\right]$$

where n is the number of the perceptron inputs, and *step* is a step activation function.

Perceptron's training algorithm (continued)

Step 3: Weight training

Update the weights of the perceptron

$$w_i(p+1) = w_i(p) + \Delta w_i(p)$$

where $\Delta w_i(p)$ is the weight correction at iteration p.

The weight correction is computed by the **delta rule**:

$$\Delta w_i(p) = \alpha \cdot x_i(p) \cdot e(p)$$

Step 4: Iteration

Increase iteration p by one, go back to *Step 2* and repeat the process until convergence.

Example of perceptron learning: the logical operation *AND*

Epoch	Inputs x_1	Inputs x_2	Desired output Y_d	Initial weights w_1	Initial weights w_2	Actual output Y	Error e	Final weights w_1	Final weights w_2
1	0	0	0	0.3	–0.1	0	0	0.3	–0.1
	0	1	0	0.3	–0.1	0	0	0.3	–0.1
	1	0	0	0.3	–0.1	1	–1	0.2	–0.1
	1	1	1	0.2	–0.1	0	1	0.3	0.0
2	0	0	0	0.3	0.0	0	0	0.3	0.0
	0	1	0	0.3	0.0	0	0	0.3	0.0
	1	0	0	0.3	0.0	1	–1	0.2	0.0
	1	1	1	0.2	0.0	1	0	0.2	0.0
3	0	0	0	0.2	0.0	0	0	0.2	0.0
	0	1	0	0.2	0.0	0	0	0.2	0.0
	1	0	0	0.2	0.0	1	–1	0.1	0.0
	1	1	1	0.1	0.0	0	1	0.2	0.1
4	0	0	0	0.2	0.1	0	0	0.2	0.1
	0	1	0	0.2	0.1	0	0	0.2	0.1
	1	0	0	0.2	0.1	1	–1	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1
5	0	0	0	0.1	0.1	0	0	0.1	0.1
	0	1	0	0.1	0.1	0	0	0.1	0.1
	1	0	0	0.1	0.1	0	0	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1

Threshold: $\theta = 0.2$; learning rate: $\alpha = 0.1$