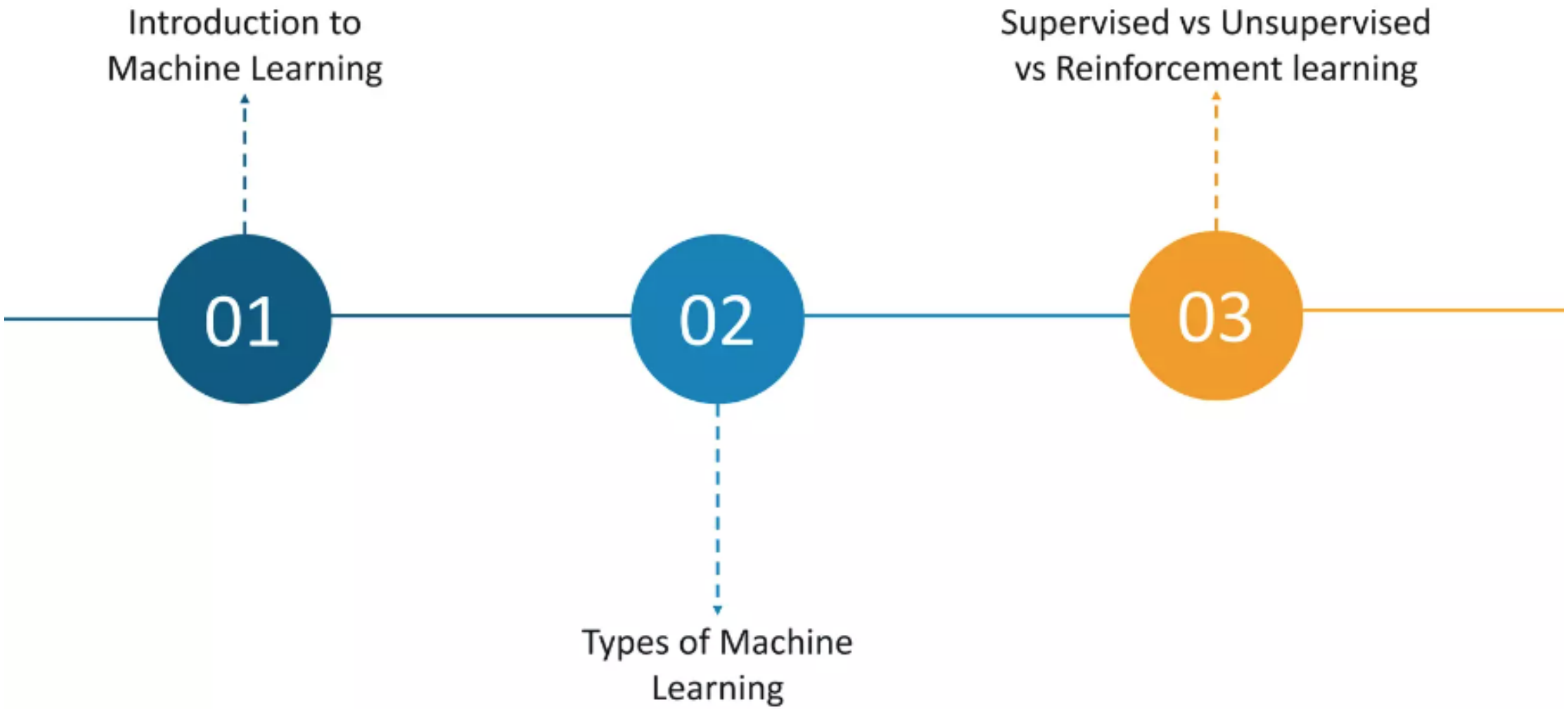
Artificial Neural Network

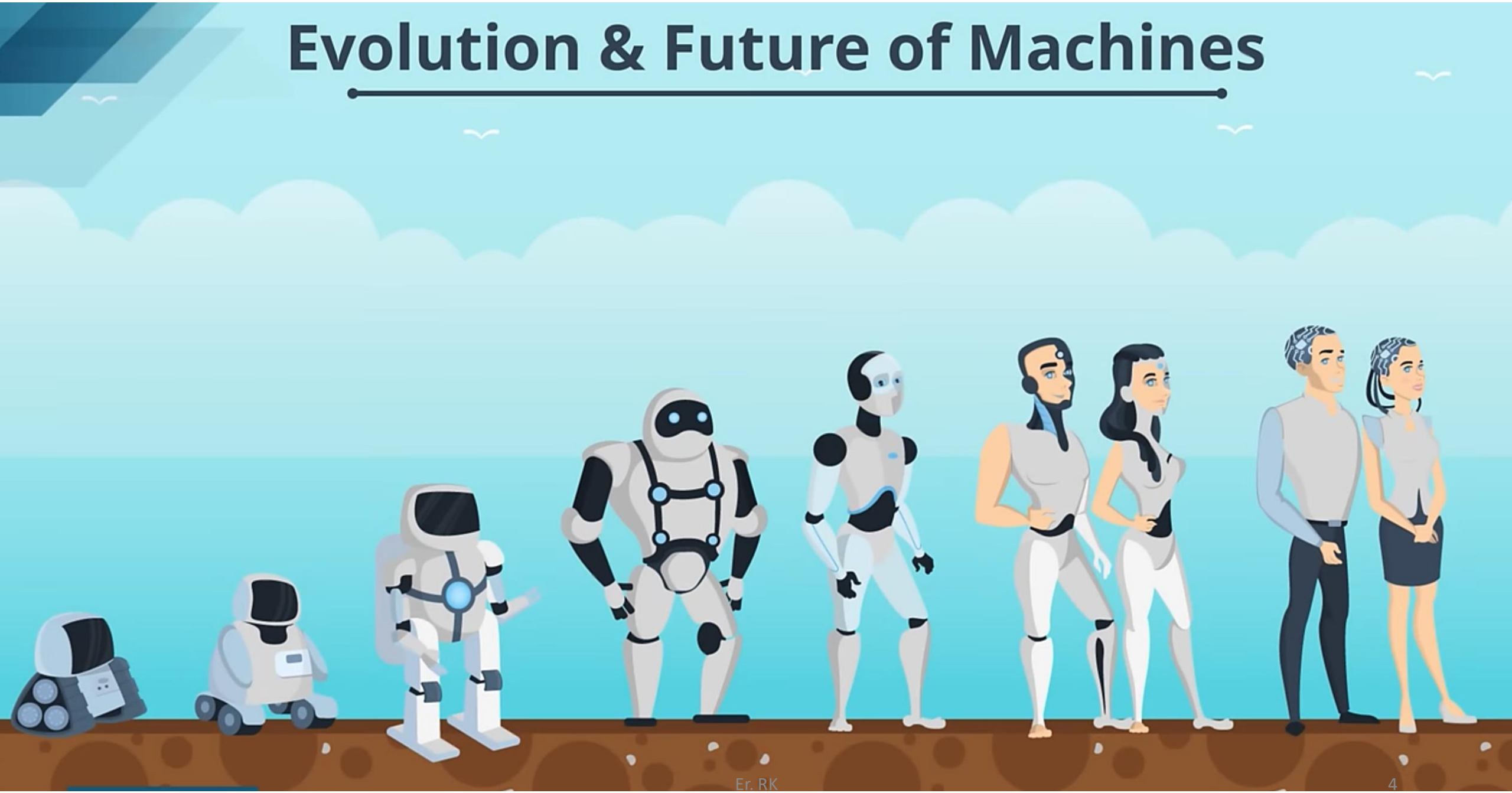
Chapter_5

Agenda


Introduction to Machine Learning
01
02
Types of Machine Learning
Supervised vs Unsupervised vs Reinforcement learning
03

Introduction To Machine Learning

Evolution & Future of Machines

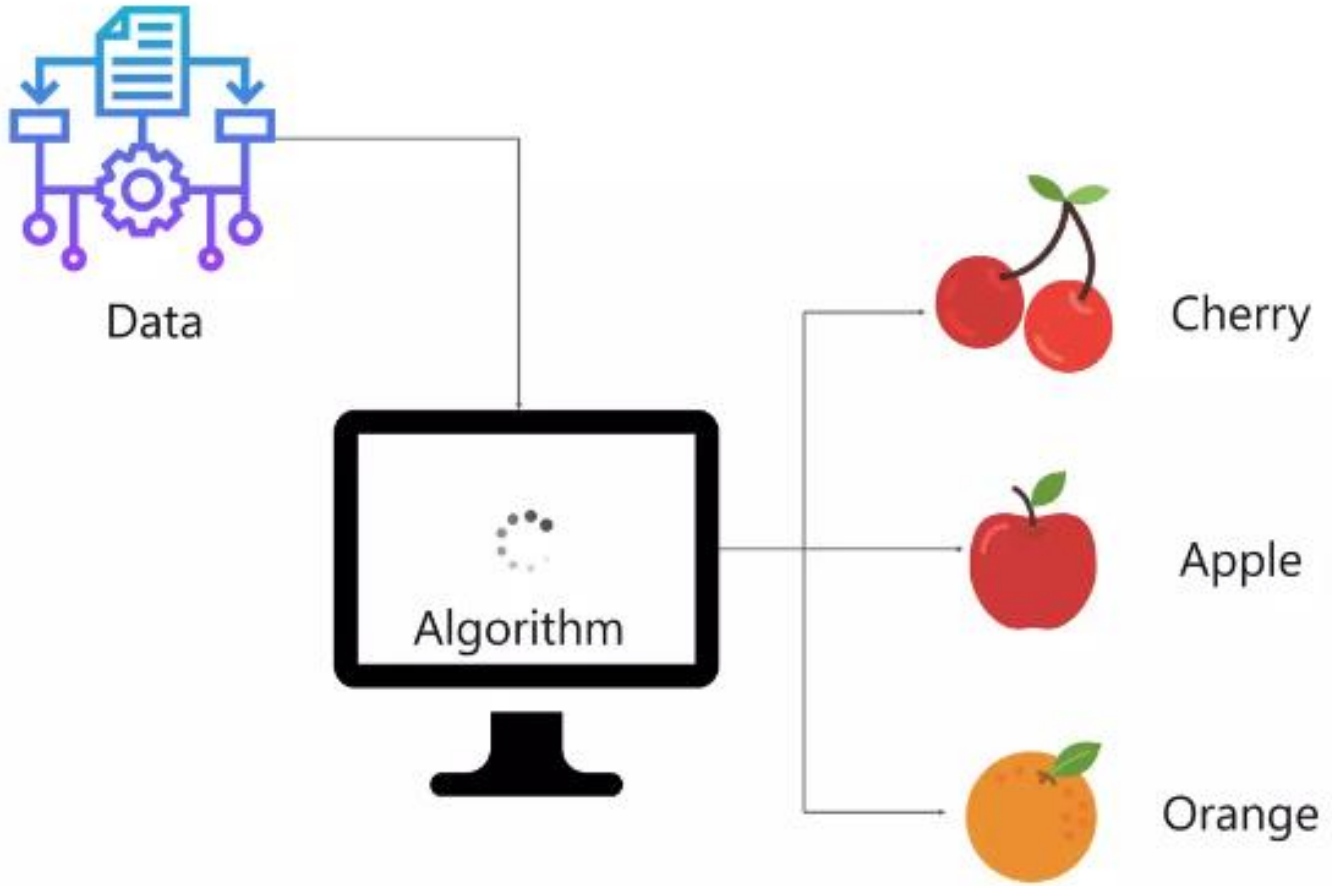


What is Machine Learning?

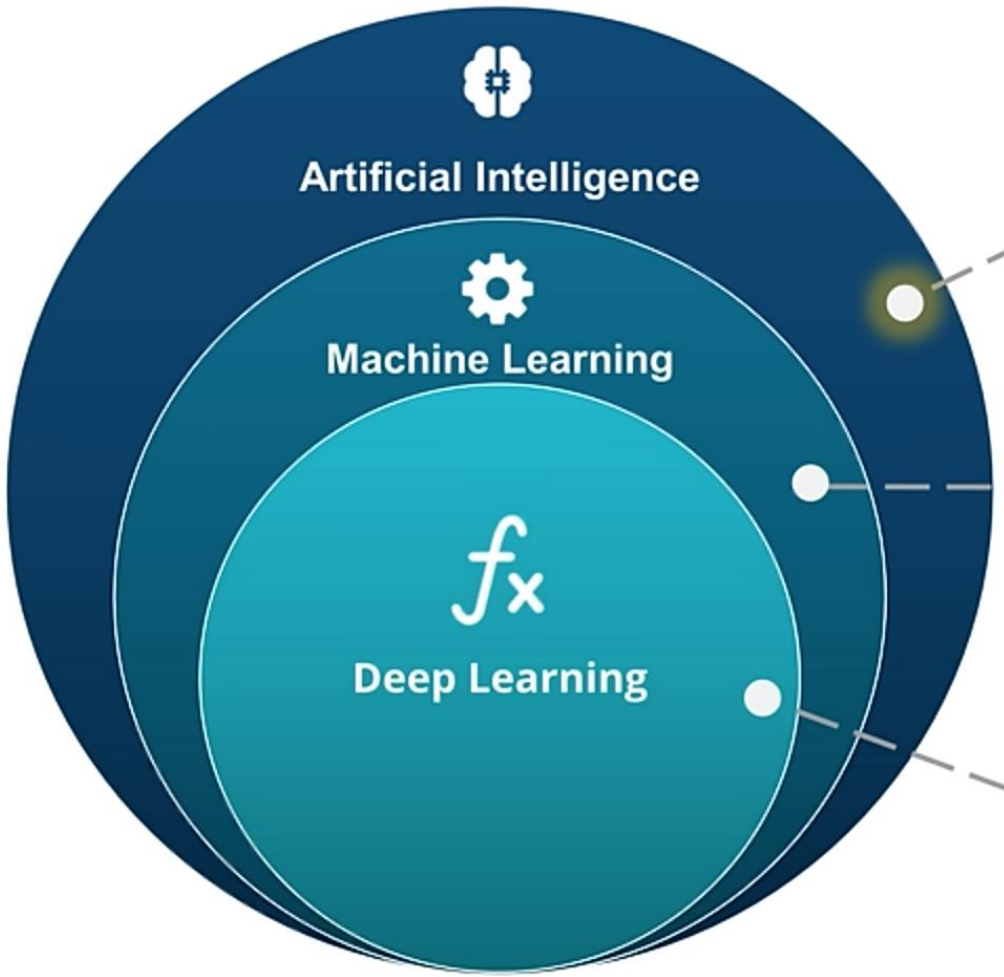
Machine Learning is a subset of artificial intelligence. It focuses mainly on the designing of systems, thereby allowing them to learn and make predictions based on some experience which is data in case of machines.

What Is Machine Learning?

Machine learning is a subset of artificial intelligence (AI) which provides machines the ability to learn automatically & improve from experience without being explicitly programmed.



Biggest Confusion: AI vs ML vs Deep Learning

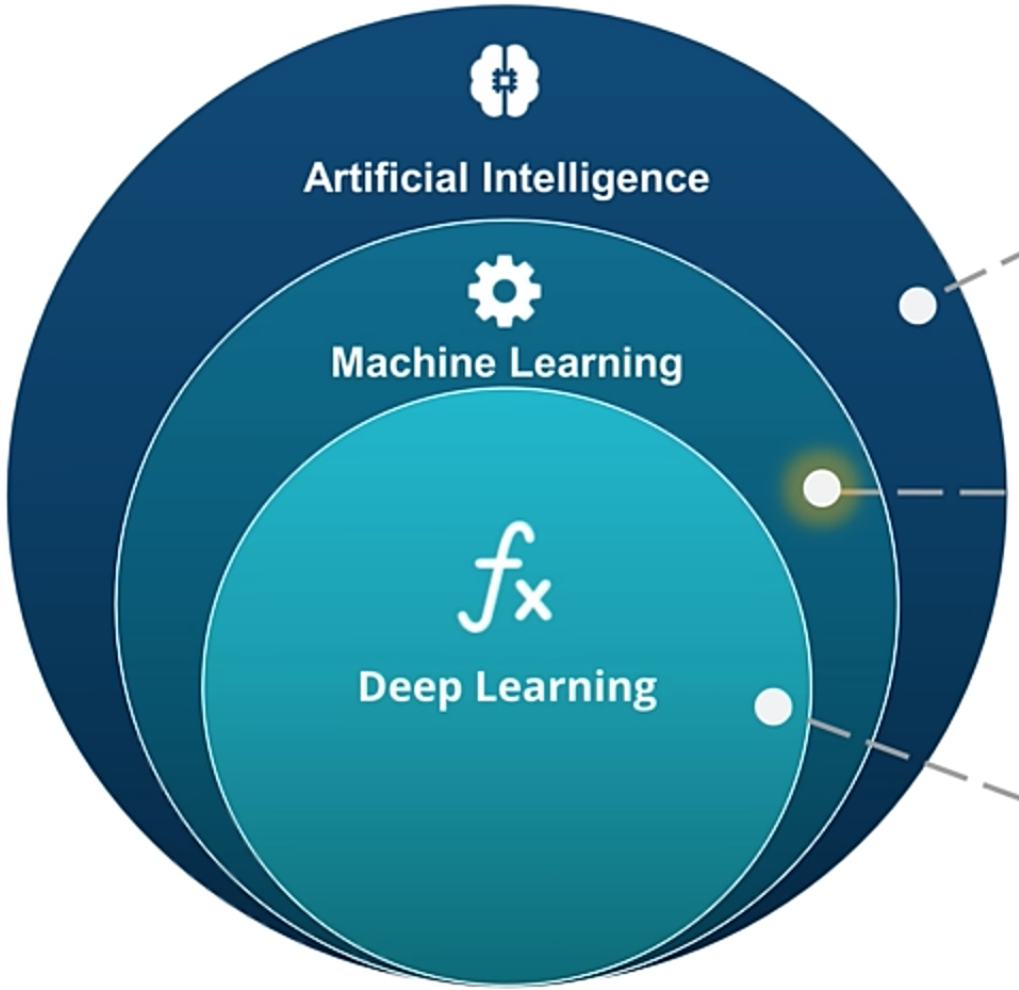


ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour

MACHINE LEARNING
Subset of AI technique which use statistical methods to enable machines to improve with experience

DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible

Biggest Confusion: AI vs ML vs Deep Learning

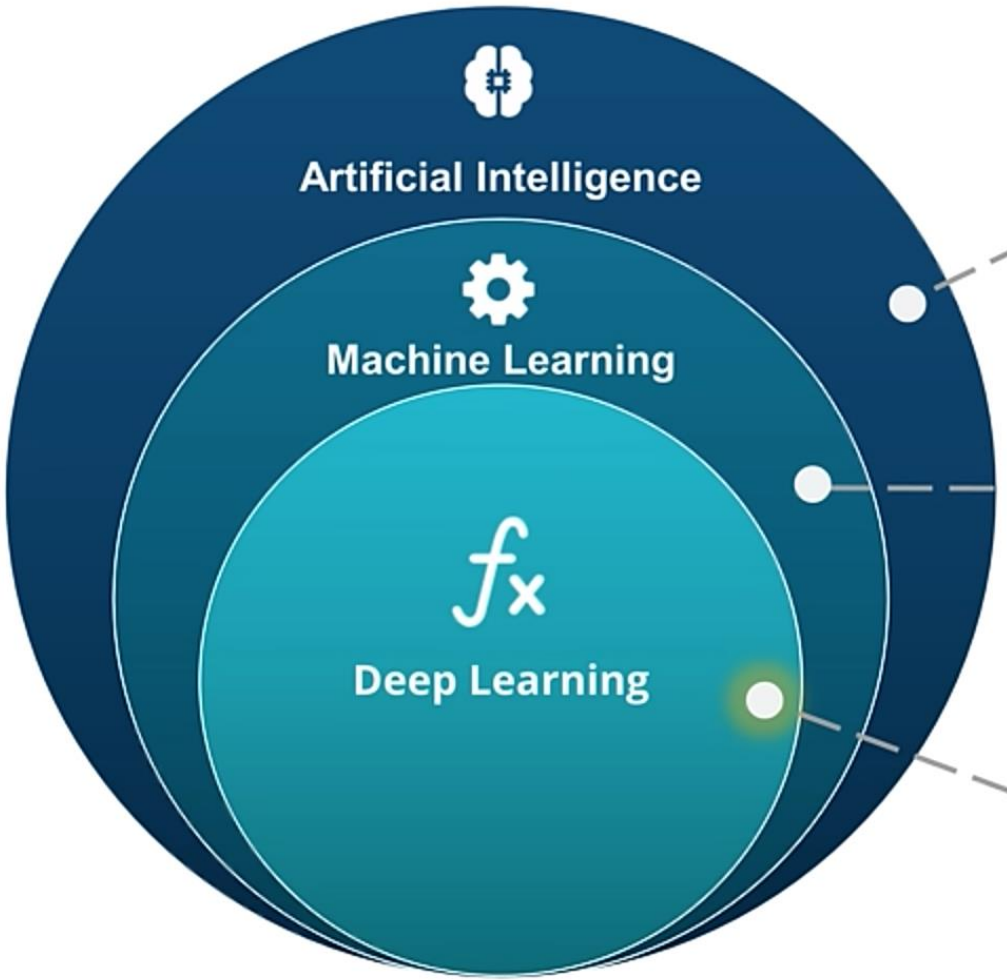


ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour

MACHINE LEARNING
Subset of AI technique which use statistical methods to enable machines to improve with experience

DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible

Biggest Confusion: AI vs ML vs Deep Learning



ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour

MACHINE LEARNING
Subset of AI technique which use statistical methods to enable machines to improve with experience

DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible

Types Of Machine Learning

Types Of Machine Learning

Supervised Learning

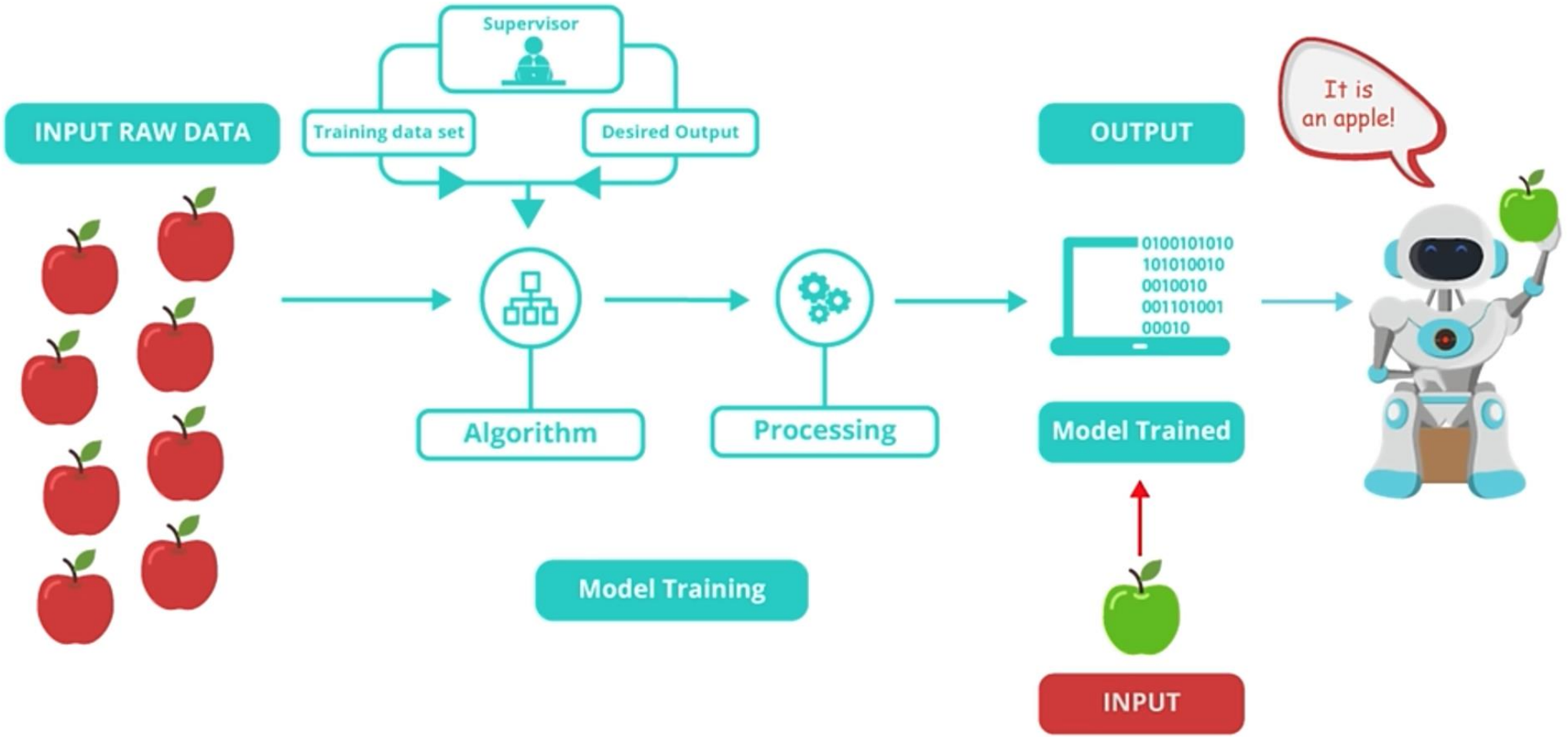
Unsupervised Learning

Reinforcement Learning

Supervised Learning

- **Definition:** Supervised Learning is a type of machine learning where the model is trained on labeled data. This means each training example is a pair consisting of an input object (typically a vector) and the desired output value (also called the supervisory signal).
- **Key Points:**
- The algorithm learns from the input-output pairs.
- The goal is to predict the output from the input.

Machine Learning: Supervised



Supervised Learning Algorithms

Linear Regression

Support Vector Machines

Random Forest

Applications of Supervised Learning

1. **Image Classification**
    1. **Example:** Identifying cats in images.
    2. **Use Case:** Social media platforms use it to tag users in photos.
2. **Spam Detection**
    1. **Example:** Filtering spam emails.
    2. **Use Case:** Email services like Gmail use it to filter out unwanted emails.
3. **Medical Diagnosis**
    1. **Example:** Predicting diseases based on patient data.
    2. **Use Case:** Hospitals use it to assist in diagnosing conditions based on symptoms and historical data.
4. **Speech Recognition**
    1. **Example:** Converting spoken words into text.
    2. **Use Case:** Virtual assistants like Siri and Alexa use it to understand user commands.
5. **Financial Fraud Detection**
    1. **Example:** Identifying fraudulent transactions.
    2. **Use Case:** Banks use it to monitor and flag suspicious activities.

Advantages of Supervised Learning

- **High Accuracy:** When labeled data is of high quality, the model can achieve high prediction accuracy.
- **Easy Interpretation:** The results and decision-making process are easier to understand and interpret.
- **Efficiency:** Can be computationally efficient with the right algorithms and resources.
- **Predictive Power:** Strong ability to predict outcomes based on new data.

Disadvantages of Supervised Learning

1. **Need for Labeled Data:** Requires a large amount of labeled data, which can be costly and time-consuming to obtain.
2. **Limited Scope:** Limited to problems where labeled data is available.
3. **Overfitting:** Models can overfit to the training data, performing poorly on new, unseen data.
4. **Time-Consuming:** Training with large datasets can be time-consuming.

Real-Time Example: Email Spam Detection

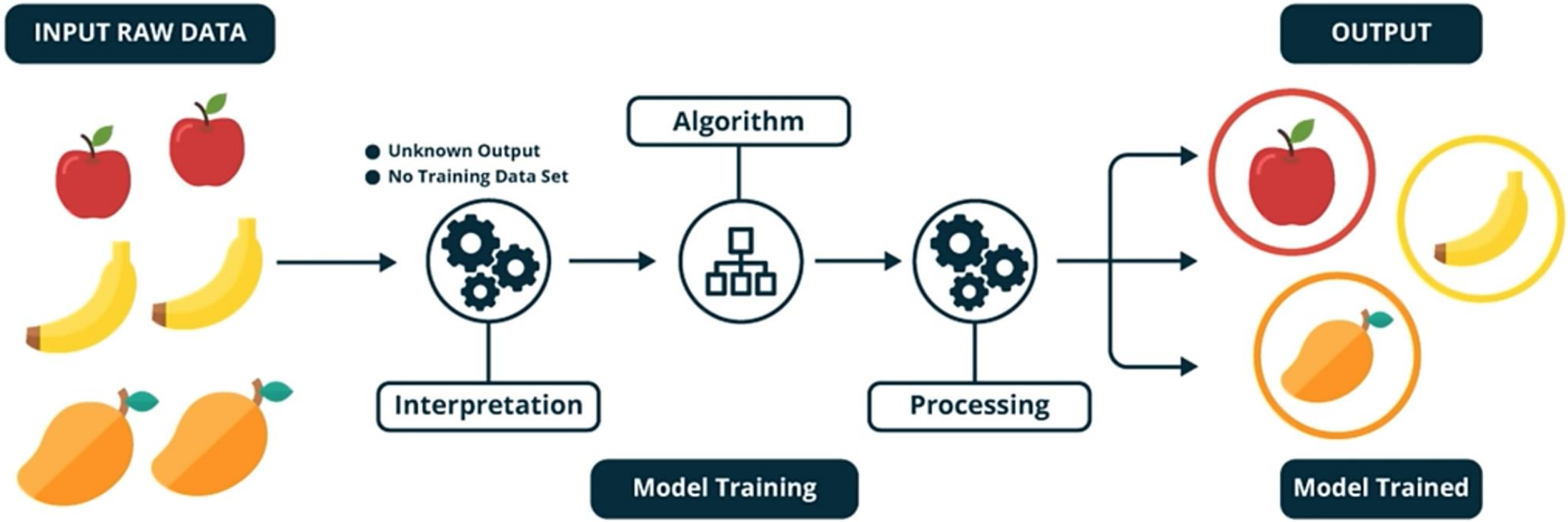
- **Description:** Email spam detection is a classic example of supervised learning. A supervised learning model can be trained on a dataset of emails labeled as 'spam' or 'not spam'. The model learns to identify features associated with spam emails and predicts whether new emails are spam.
- **Process:**

1. **Collect Data:** Gather a large dataset of emails, each labeled as spam or not spam.
2. **Feature Extraction:** Extract relevant features (e.g., keywords, sender information).
3. **Train Model:** Use supervised learning algorithms (e.g., Naive Bayes) to train the model.
4. **Predict:** Use the trained model to classify new emails.

Unsupervised Learning

- **Definition:** Unsupervised Learning is a type of machine learning where the model is trained on unlabeled data. The algorithm tries to find patterns and relationships in the data without explicit instructions on what to look for.
- **Key Points:**
- **No Labels:** The data provided to the model is not labeled.
- **Pattern Recognition:** The goal is to identify hidden patterns and structures in the data.

Machine Learning: Unsupervised



Key Goals of Unsupervised Learning:

1. **Clustering** — Group similar data points together.
2. **Dimensionality Reduction** — Simplify data by reducing the number of features while retaining essential information.
3. **Anomaly Detection** — Identify unusual patterns or outliers.

Types of Unsupervised Learning:

Apriori Algorithm

K- Means Algorithm

Hierarchical Clustering

- **1. Clustering:**
- Dividing data into groups (clusters) where members of the same group are more similar to each other.
- **Examples:**
  - Customer segmentation in marketing.
  - Document categorization.
  - Social network analysis.
- **Popular Clustering Algorithms:**
- **K-Means Clustering**
- **Hierarchical Clustering**
- **DBSCAN (Density-Based Spatial Clustering)**

- **2. Dimensionality Reduction:**
- Reducing the number of input variables (features) to simplify models and visualize data.
- **Examples:**
  - Visualizing high-dimensional data (e.g., PCA reduces to 2D/3D).
  - Noise removal.
- **Popular Techniques:**
- **PCA (Principal Component Analysis)**
- **t-SNE (t-distributed Stochastic Neighbor Embedding)**
- **Autoencoders (Neural Network-based)**

Applications of Unsupervised Learning

- **Clustering**
- **Example:** Grouping customers based on purchasing behavior.
- **Use Case:** E-commerce platforms use it to segment customers for targeted marketing.
- **Anomaly Detection**
- **Example:** Identifying unusual transactions in banking.
- **Use Case:** Banks use it to detect fraudulent activities.
- **Market Basket Analysis**
- **Example:** Finding associations between products in a shopping cart.
- **Use Case:** Retailers use it to create product bundles and improve cross-selling.
- **Dimensionality Reduction**
- **Example:** Reducing the number of features in a dataset while preserving important information.
- **Use Case:** Used in pre-processing for machine learning tasks to improve model performance and reduce computational cost.
- **Recommendation Systems**
- **Example:** Suggesting movies based on user viewing history.
- **Use Case:** Streaming services like Netflix and Spotify use it to recommend content to users.

Advantages of Unsupervised Learning

1. **No Need for Labeled Data**

    - **Explanation:** Unsupervised learning can work with unlabeled data, which is often more readily available and cheaper to obtain.
    - **Impact:** This makes it suitable for applications where labeling data is infeasible or too expensive.

2. **Discover Hidden Patterns**

    - **Explanation:** It can uncover hidden patterns and structures in the data that may not be immediately apparent.
    - **Impact:** This capability is valuable for exploratory data analysis and gaining insights from data.

Advantages of Unsupervised Learning

3. **Flexibility**

    - **Explanation:** Unsupervised learning can adapt to new and changing data without the need for retraining with labeled data.
    - **Impact:** This makes it suitable for dynamic environments where data evolves over time.

4. **Pre-processing Tool**

    - **Explanation:** Techniques like clustering and dimensionality reduction are often used to preprocess data for other machine learning tasks.
    - **Impact:** This can improve the performance and efficiency of supervised learning models.

Disadvantages of Unsupervised Learning

1. **Complexity in Interpretation**

    - **Explanation:** The results of unsupervised learning can be harder to interpret and validate compared to supervised learning.
    - **Impact:** This can make it challenging to explain the findings to stakeholders or integrate them into decision-making processes.

2. **Evaluation Challenges**

    - **Explanation:** Evaluating the performance of unsupervised learning models is more difficult because there are no labeled outcomes to compare against.
    - **Impact:** This can lead to uncertainty about the model's accuracy and reliability.

Disadvantages of Unsupervised Learning

3. **Unpredictability**

   - **Explanation:** The patterns and structures discovered by unsupervised learning may not always be relevant or useful for the intended application.
   - **Impact:** This unpredictability can result in models that do not meet the desired objectives.

4. **Computationally Intensive**

   - **Explanation:** Some unsupervised learning algorithms, like clustering on large datasets, can be computationally intensive and require significant resources.
   - **Impact:** This can limit their use in real-time applications or on devices with limited processing power.

Real-Time Example: Customer Segmentation in E-commerce

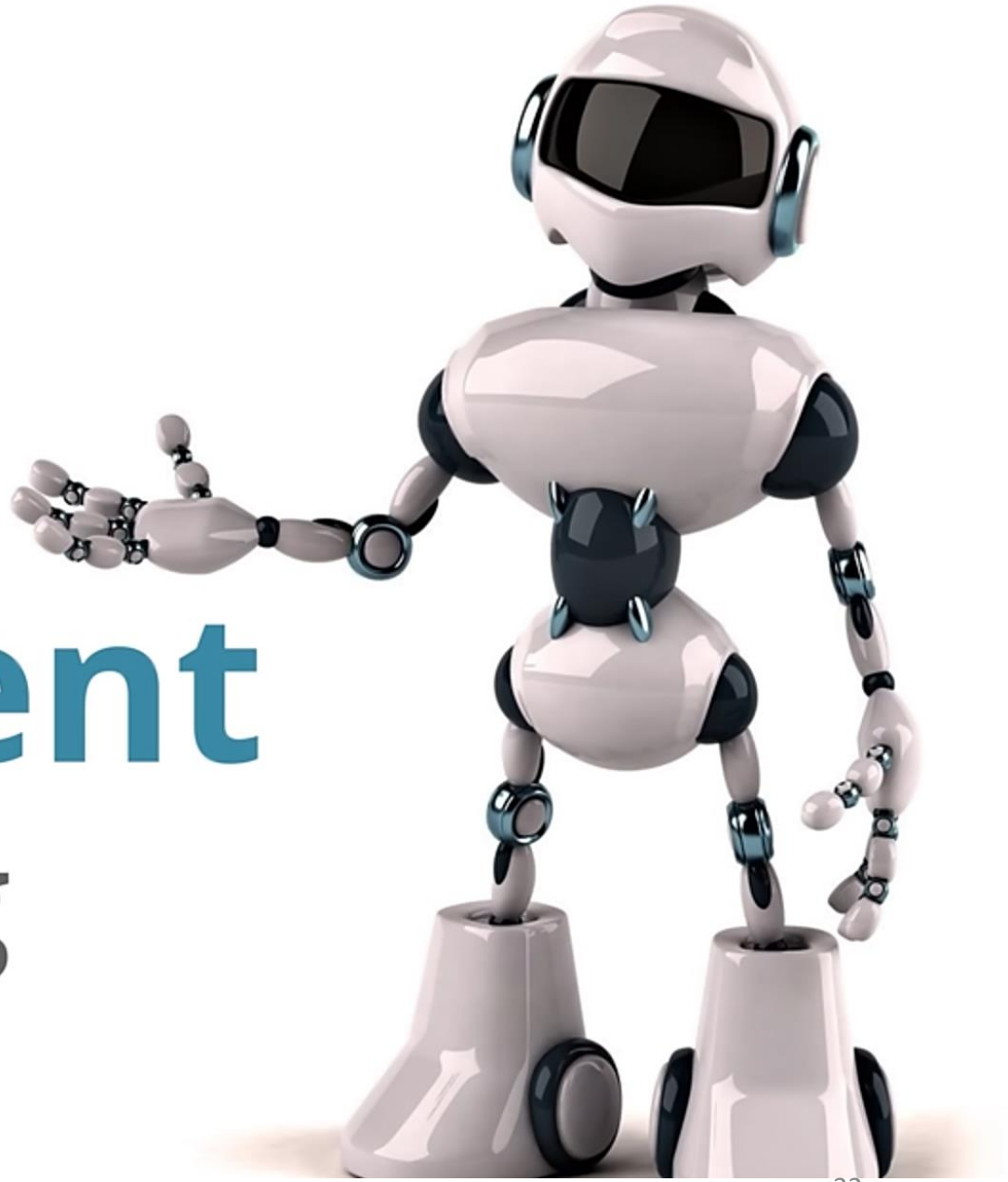
- **Description:** Customer segmentation is a common application of unsupervised learning. An e-commerce platform can use clustering algorithms to group customers based on their purchasing behavior, preferences, and browsing patterns.
- **Process:**

1. **Collect Data:** Gather data on customer purchases, browsing history, demographics, etc.
2. **Feature Extraction:** Identify relevant features that describe customer behavior.
3. **Apply Clustering Algorithm:** Use clustering algorithms like K-means to group customers into segments.
4. **Analyze Segments:** Analyze the characteristics of each segment to understand customer needs and preferences.

Real-Time Example: Customer Segmentation in E-commerce

- **Benefits:**
- **Targeted Marketing:** Enables personalized marketing campaigns tailored to different customer segments.
- **Product Recommendations:** Helps in recommending products that are more likely to be of interest to each segment.
- **Improved Customer Satisfaction:** Enhances the overall customer experience by providing more relevant offers and services.

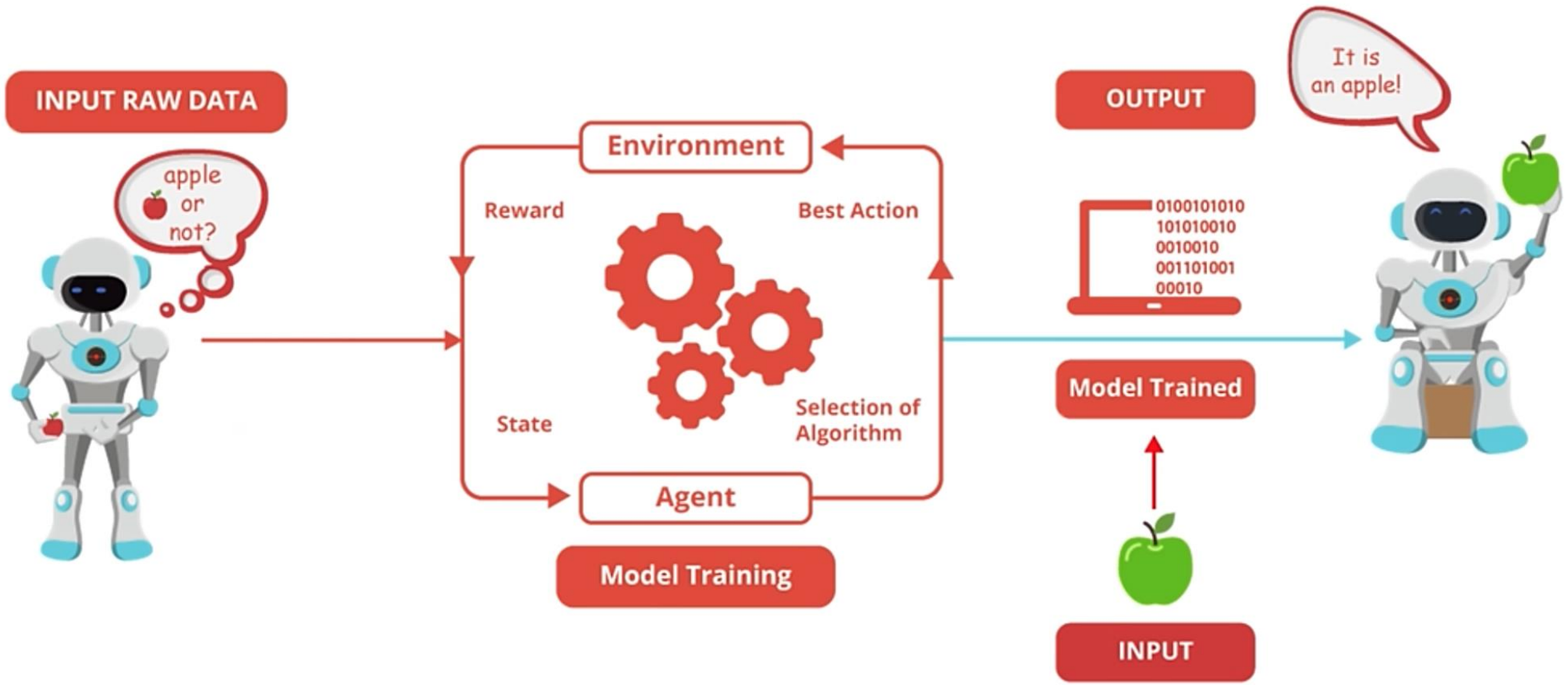
Reinforcement
Machine Learning



Reinforcement Learning

- **Definition:** Reinforcement Learning (RL) is a type of machine learning where an agent learns to make decisions by performing actions in an environment to maximize cumulative reward. The agent receives feedback in the form of rewards or penalties based on the actions it takes, allowing it to learn the best strategy over time.
- **Key Points:**
- **Agent:** The learner or decision-maker.
- **Environment:** The context in which the agent operates.
- **Actions:** Choices made by the agent.
- **Rewards:** Feedback from the environment to evaluate the actions.

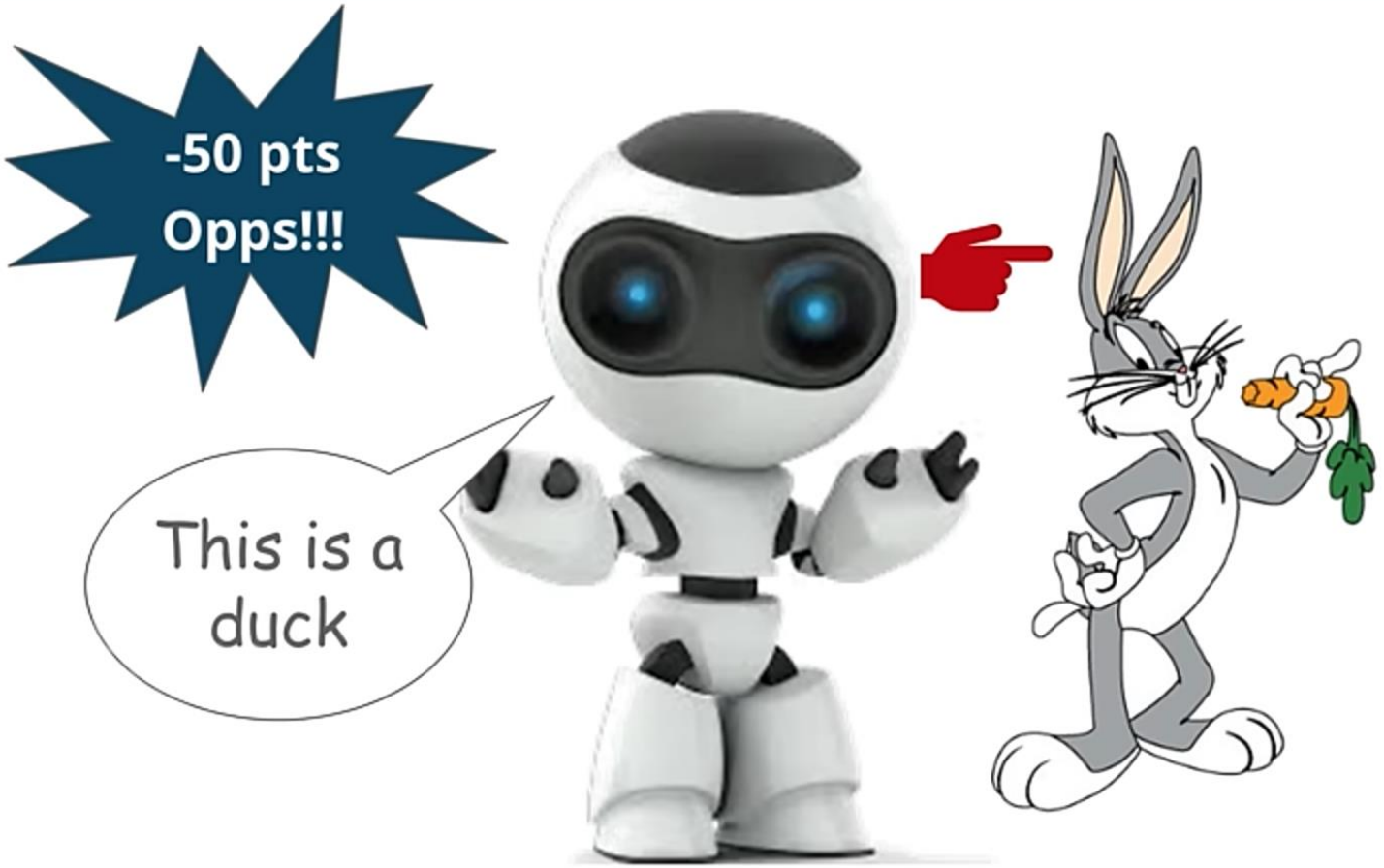
Machine Learning: Reinforcement



Machine Learning: Reinforcement

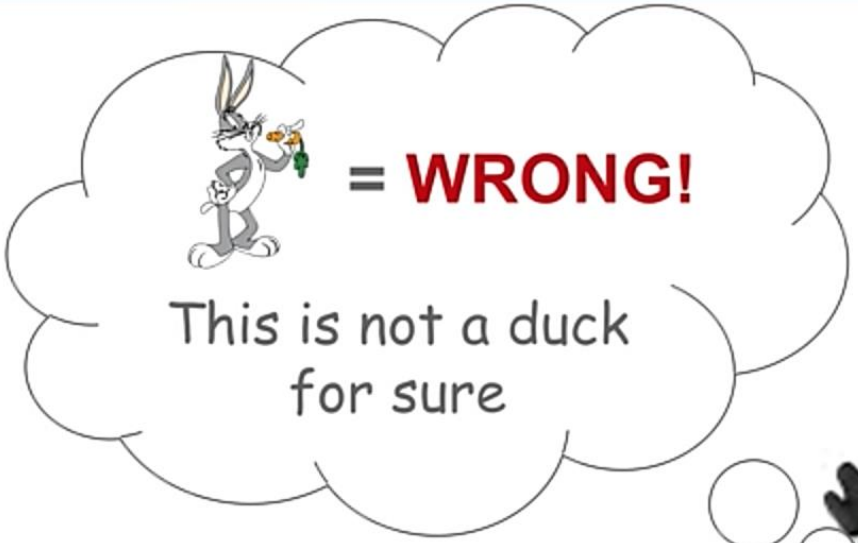
1. Observe
2. Select Action Using Policy

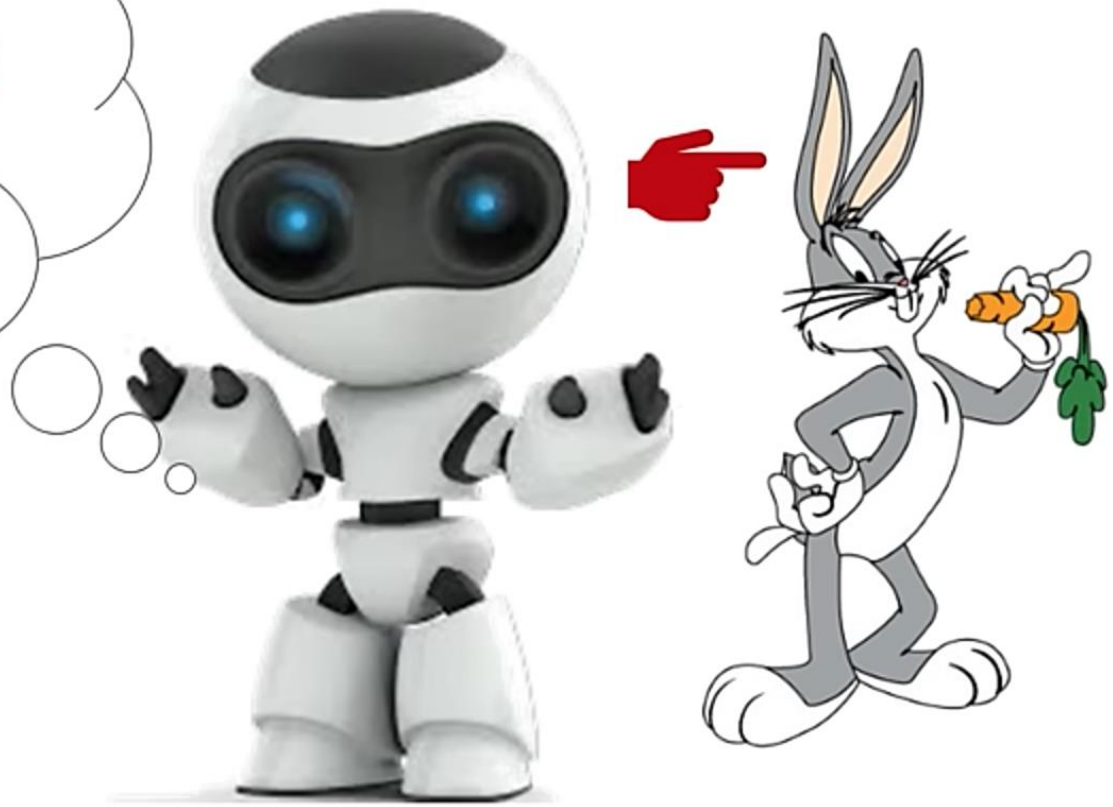
Machine Learning: Reinforcement



3. Action!
4. Get Reward or Penalty

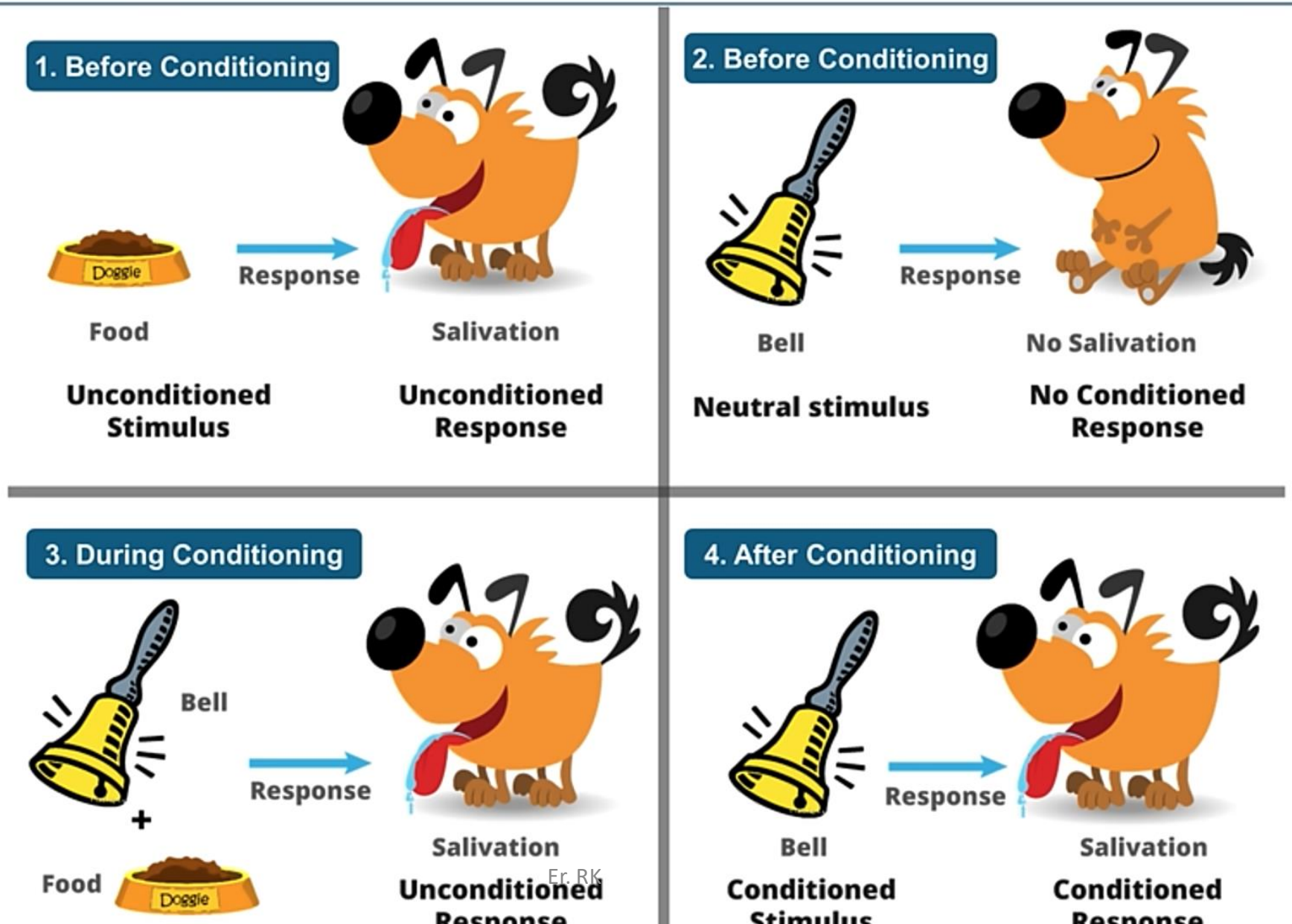
Machine Learning: Reinforcement





5. Update Policy (learning step)
6. Iterate to get Optimal Policy

Reinforcement Training with Dog



Applications of Reinforcement Learning

- **Robotics**
- **Example:** Teaching robots to navigate and perform tasks.
- **Use Case:** Autonomous robots in manufacturing and healthcare for tasks like assembly and patient assistance.
- **Game Playing**
- **Example:** Developing agents that can play and master games.
- **Use Case:** AI playing and winning complex games like Go and Chess (e.g., AlphaGo by DeepMind).
- **Autonomous Vehicles**
- **Example:** Enabling self-driving cars to make real-time decisions.
- **Use Case:** Autonomous vehicles navigating roads, avoiding obstacles, and optimizing routes.

Applications of Reinforcement Learning

- **Financial Trading**
- **Example:** Creating algorithms for automated trading.
- **Use Case:** High-frequency trading systems that make decisions to buy or sell stocks based on market conditions.
- **Personalized Recommendations**
- **Example:** Improving recommendation systems based on user interaction.
- **Use Case:** Online platforms like Netflix and Amazon suggesting movies and products to users.

Advantages of Reinforcement Learning

- **Dynamic Learning**
- **Explanation:** RL algorithms can adapt to changing environments and learn optimal strategies over time.
- **Impact:** This makes them suitable for applications requiring continuous learning and adaptation, like autonomous driving.
- **High Efficiency**
- **Explanation:** RL can find solutions that are not evident through traditional programming by exploring and exploiting the environment.
- **Impact:** This efficiency is beneficial in complex problem-solving tasks, such as game playing.

Advantages of Reinforcement Learning

- **Scalability**
- **Explanation:** RL can be applied to a wide range of problems, from simple tasks to complex, multi-step processes.
- **Impact:** This scalability makes it versatile and applicable across various domains.
- **Automation of Complex Tasks**
- **Explanation:** RL can automate tasks that are challenging to model explicitly.
- **Impact:** This leads to the automation of sophisticated processes, such as robotic manipulation and financial trading.

Disadvantages of Reinforcement Learning

- **High Computational Cost**
- **Explanation:** RL algorithms often require significant computational resources and time to train.
- **Impact:** This can be a barrier to implementation, especially for large-scale or real-time applications.
- **Complexity in Implementation**
- **Explanation:** Designing and tuning RL algorithms can be complex and requires expertise.
- **Impact:** This complexity can slow down the development process and increase the need for specialized skills.

Disadvantages of Reinforcement Learning

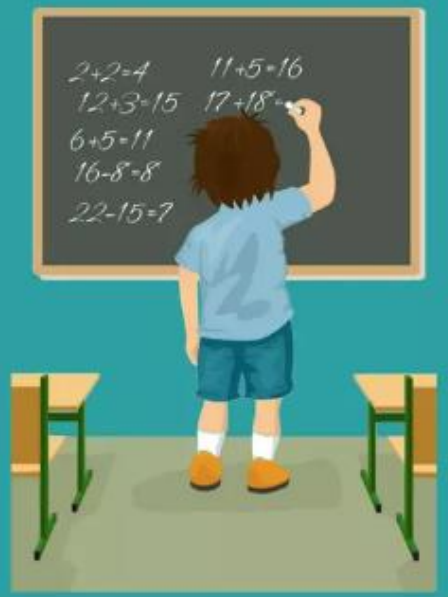
- **Need for Extensive Training**
- **Explanation:** RL often needs a large amount of interaction data with the environment to learn effectively.
- **Impact:** This extensive training can be impractical for some applications and environments.
- **Risk of Suboptimal Policies**
- **Explanation:** RL agents may converge to local optima rather than global optima, leading to suboptimal policies.
- **Impact:** This can result in less effective solutions, especially in highly complex environments.

Supervised vs Unsupervised vs Reinforcement

Definition

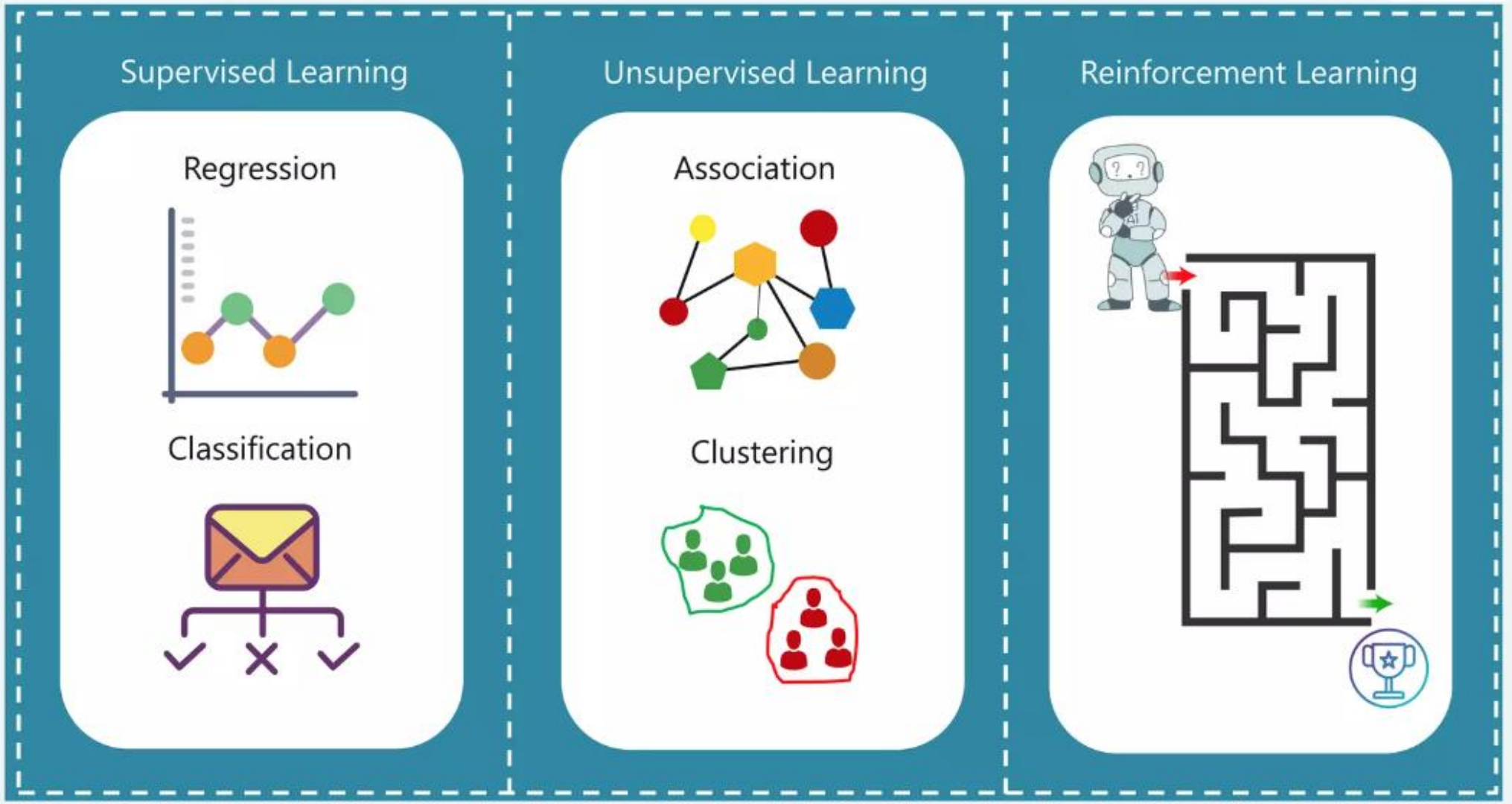
Supervised learning is a method in which we teach the machine using labelled data

In unsupervised learning the machine is trained on unlabelled data without any guidance



In Reinforcement learning an agent interacts with its environment by producing actions & discovers errors or rewards

Problem Type

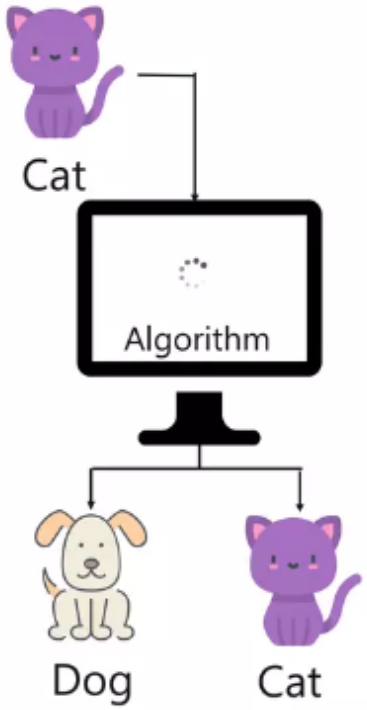


Type of data

- Definition
- Type of Problems
- Type of data
- Training
- Aim
- Approach
- Output Feedback
- Popular Algorithms
- Applications

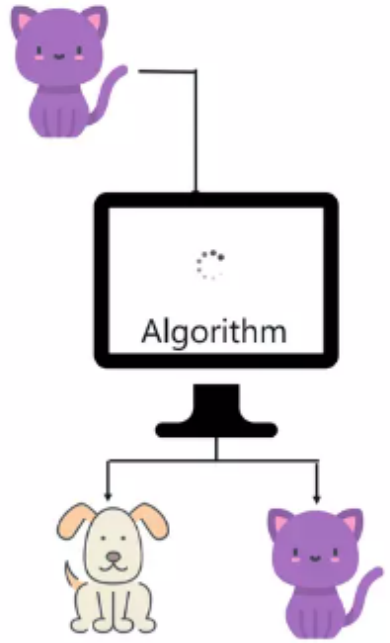
Supervised Learning

Labelled Data



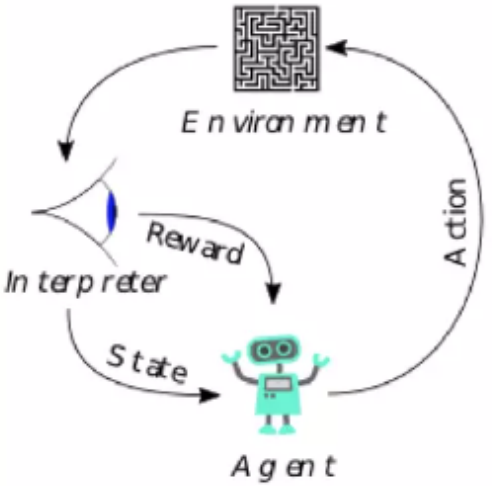
Unsupervised Learning

Unlabelled Data

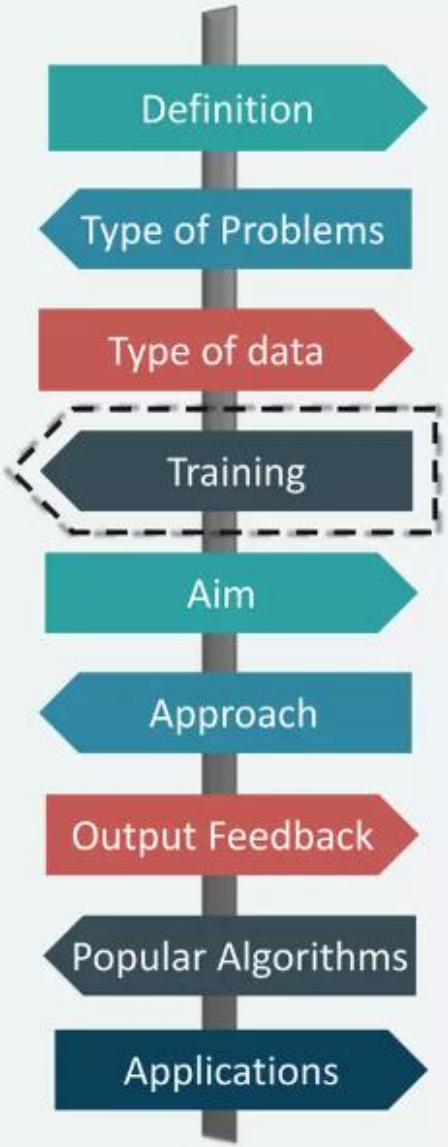


Reinforcement Learning

No Predefined Data



Training



Supervised Learning

External supervision

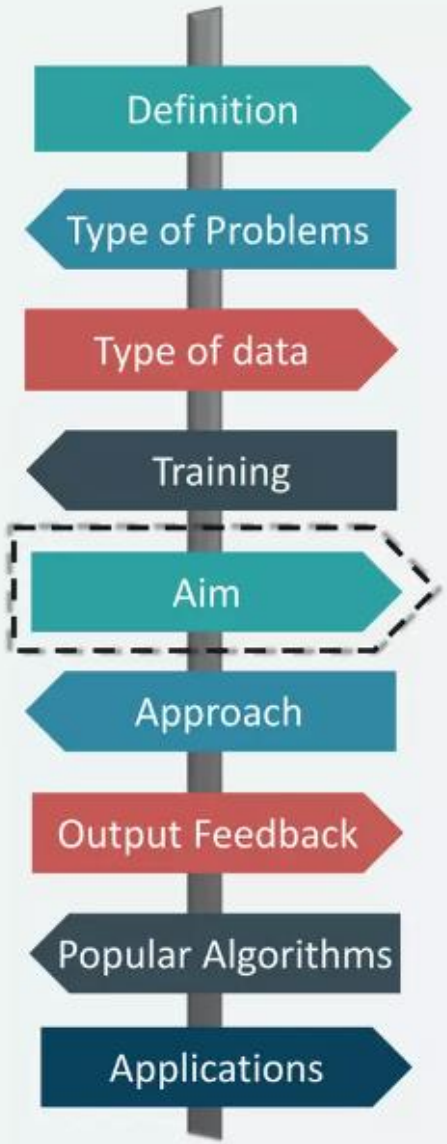
Unsupervised Learning

No supervision

Reinforcement Learning

No supervision

Aim



Supervised Learning

Forecast outcomes

Unsupervised Learning

Discover underlying patterns

Reinforcement Learning

Learn series of action

Approach

- Definition
- Type of Problems
- Type of data
- Training
- Aim
- Approach
- Output Feedback
- Popular Algorithms
- Applications

Supervised Learning

Map labelled input to known output

Labelled Input → Training → Algorithm → Known Output

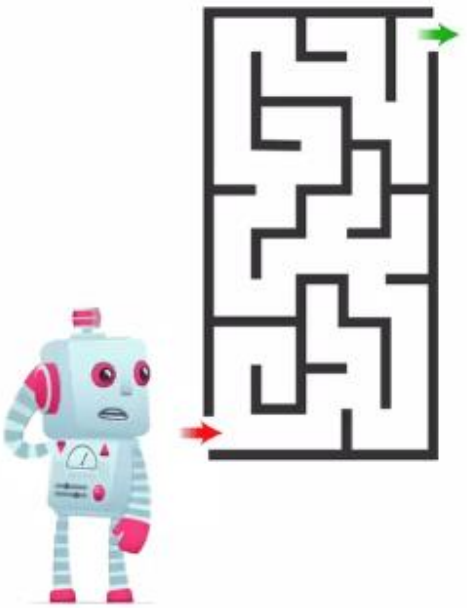
Unsupervised Learning

Understand patterns and discover output

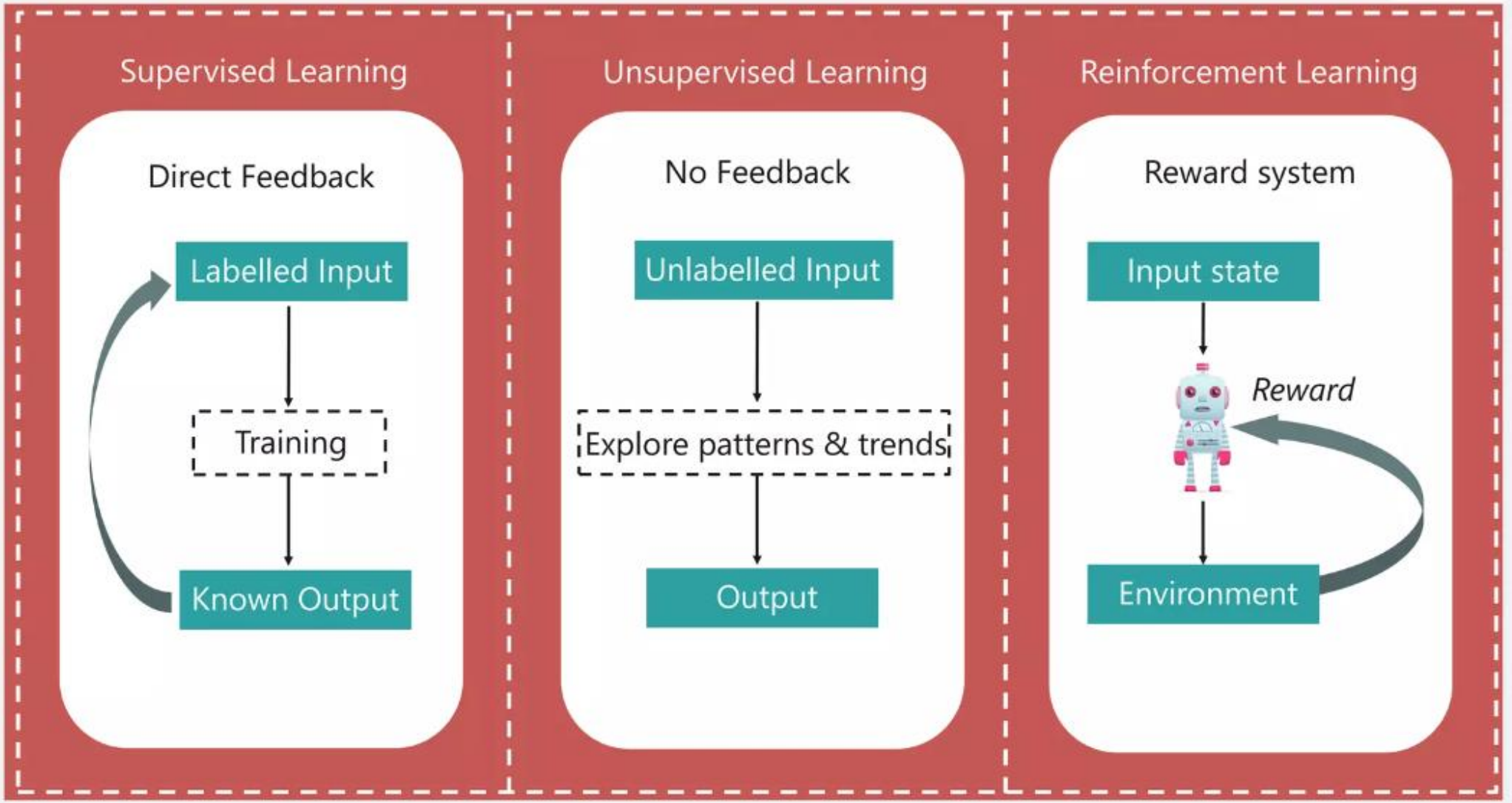
Unlabelled Input → Explore patterns & trends → Algorithm → Output

Reinforcement Learning

Follow Trail and Error method



Output Feedback



Popular Algorithms

Supervised Learning

- Linear Regression
- Logistic Regression
- Support Vector Machine
- K Nearest Neighbour
- Random Forest

Unsupervised Learning

- K- Means
- Apriori
- C- Means

Reinforcement Learning

- Q- Learning
- SARSA

Applications

Supervised Learning

Risk Evaluation

Forecast Sales

Unsupervised Learning

Recommendation Systems

Anomaly Detection

Reinforcement Learning

Self driving cars

Gaming

Learning with Neural Networks

Neural Networks

- Neural networks are **computational models inspired by the human brain**, used for machine learning tasks like classification, regression, pattern recognition, etc.
- When we say *learning with neural networks*, we mean training the network to adjust its internal parameters (weights and biases) so that it performs a desired task well — by minimizing an error function (loss).

Biological Neural Network

- A **biological neural network (BNN)** refers to the intricate web of **neurons** and **connections** in the nervous system — primarily the **brain** — that enables living organisms to perceive, think, learn, and act.
  - Composed of **billions of neurons**, each connected to thousands of others.
  - These neurons communicate via **electrochemical signals**.
  - Learning occurs when the strength of these connections (synapses) is modified — called **synaptic plasticity**.

Basic Components of a BNN

Component	Function
Neuron (nerve cell)	The basic building block of the nervous system; processes and transmits information.
Dendrites	Branch-like structures that receive signals from other neurons.
Cell Body (Soma)	Contains the nucleus and integrates incoming signals.
Axon	Long projection that carries electrical impulses away from the soma to other neurons or muscles.
Synapse	The tiny gap between the axon of one neuron and the dendrite of another where signals are transmitted chemically.

How Neurons Work

- **Step-by-Step:**
- A neuron receives signals through its **dendrites**
- These signals are summed in the **soma**.
- If the total signal exceeds a **threshold**, the neuron “fires” an **action potential**.
- The action potential travels along the **axon** to the synapse.
- At the **synapse**, it releases **neurotransmitters**, which influence the next neuron.

Key Properties of BNNs:

1. Parallelism:

- Millions of neurons fire simultaneously.
- Enables massive parallel processing.

2. Plasticity:

- Synapses can strengthen or weaken depending on use.
- This is how learning and memory occur biologically.

3. Fault Tolerance:

- If some neurons die, others often compensate.
- Robust and self-repairing to some extent.

4. Energy Efficiency:

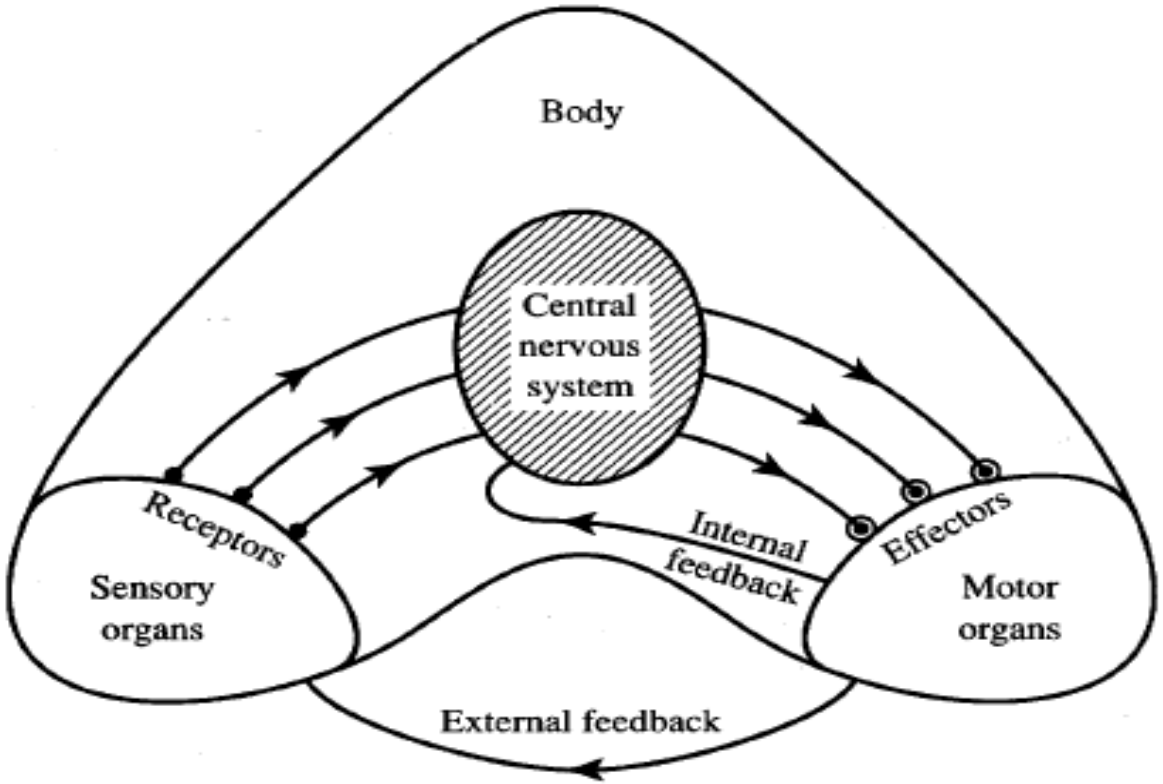
- Despite its complexity, the human brain runs on ~20 watts — much more efficient than computers.

Example: Biological Neural Circuit

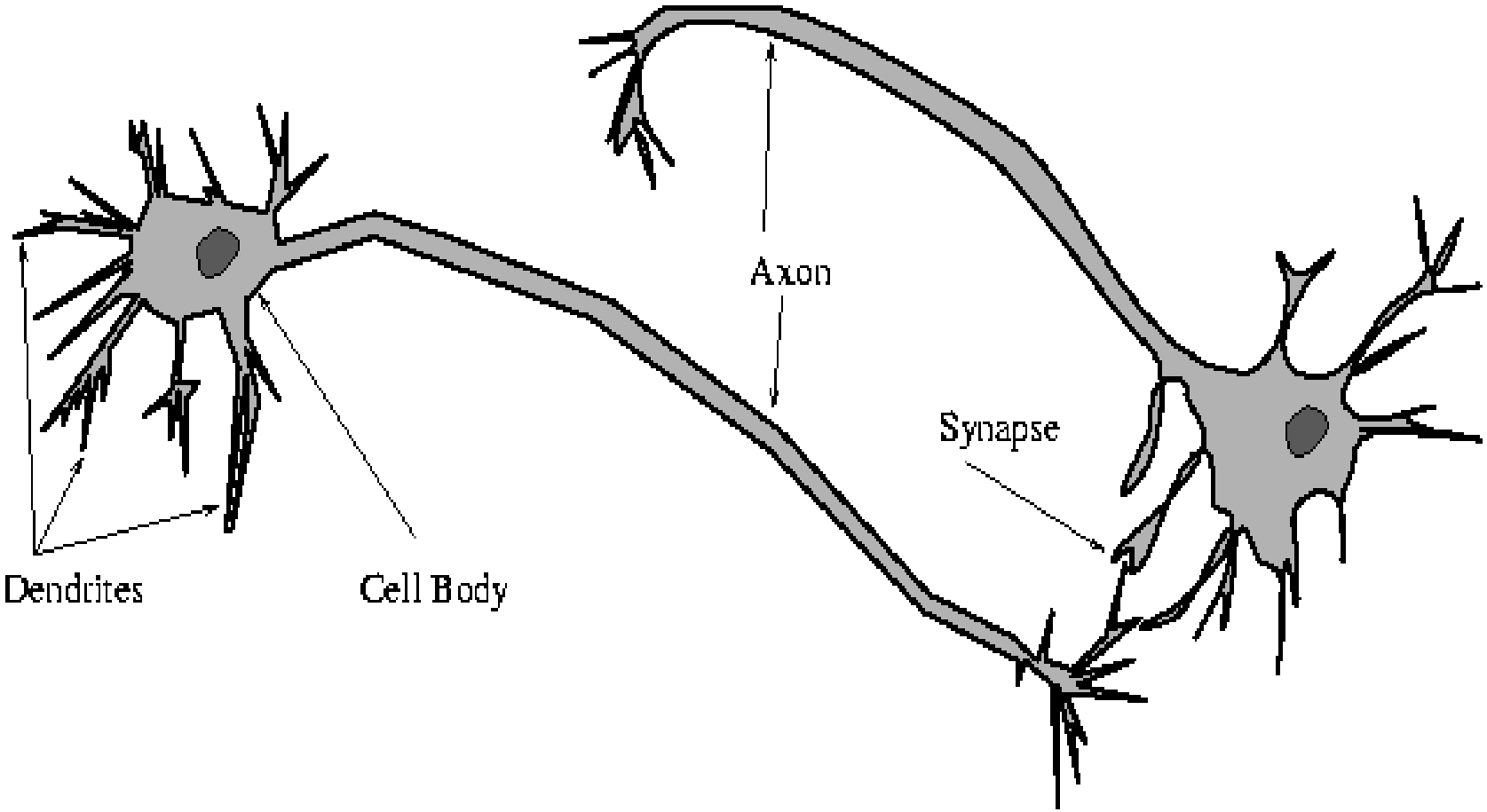
- Consider **reflex action**:
  - Sensory neuron senses pain → sends signal to spinal cord.
  - Interneurons in spinal cord process the signal.
  - Motor neurons trigger muscle contraction → move hand away.

Artificial Neural Network

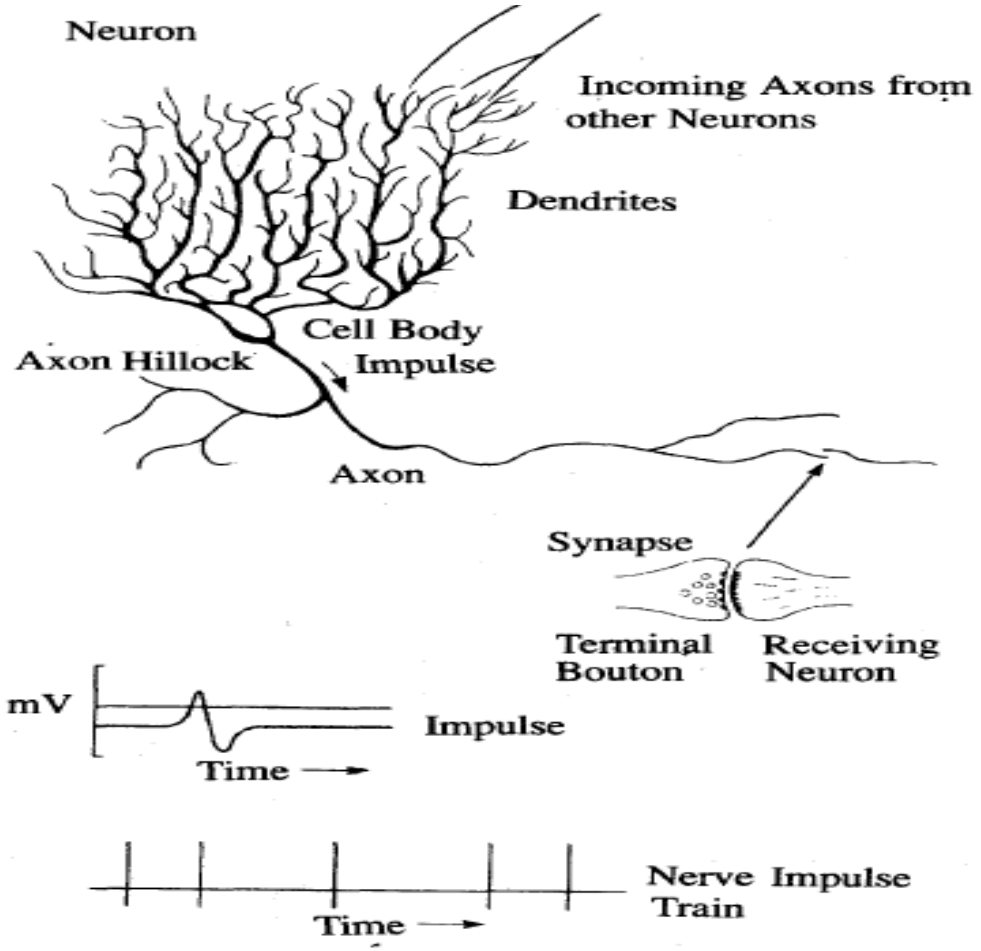
Information flow in nervous system



Biological Neural Network



Neuron and a sample of pulse train



ANN

- An **ANN** is a **computational model** inspired by the structure & function of the human brain.
  It's a **collection of artificial neurons (nodes)** arranged in layers and connected via **weights**.
- ANNs can **learn patterns, approximate functions, and make predictions** from data by adjusting these weights.

How does an ANN work?

- At a high level:
  Data flows from **input layer** → through **hidden layers** → to **output layer**.
  Each neuron:
  - Takes weighted sum of inputs
  - Adds bias
  - Passes through activation function (like a threshold)
    ✅ The model adjusts its internal weights & biases during training to minimize error.

Basic Building Block: Artificial Neuron

Mathematical Model:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

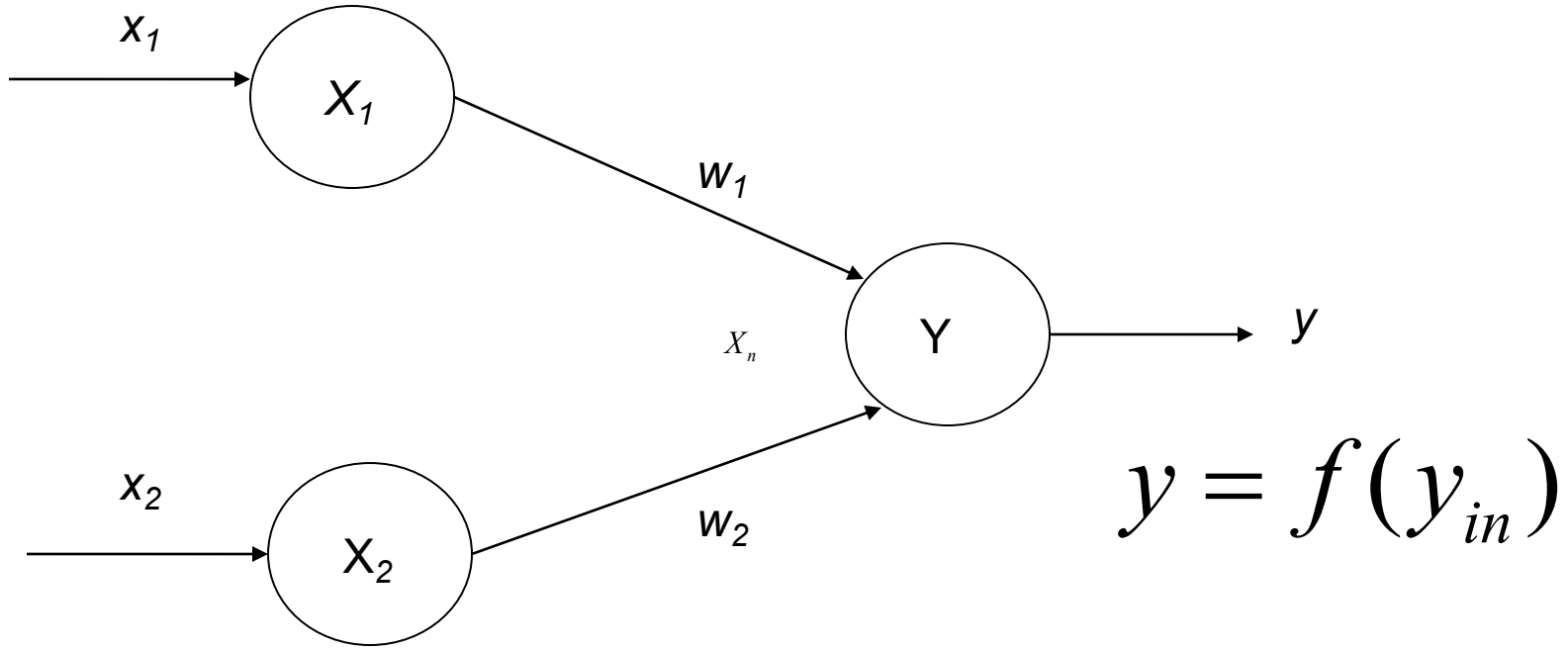
Where:

- x_i: input values
- w_i: weights
- b: bias
- f: activation function
- y: output

Architecture of an ANN

Layer	Function
Input layer	Receives raw data
Hidden layer(s)	Learns features, performs computations
Output layer	Produces final prediction

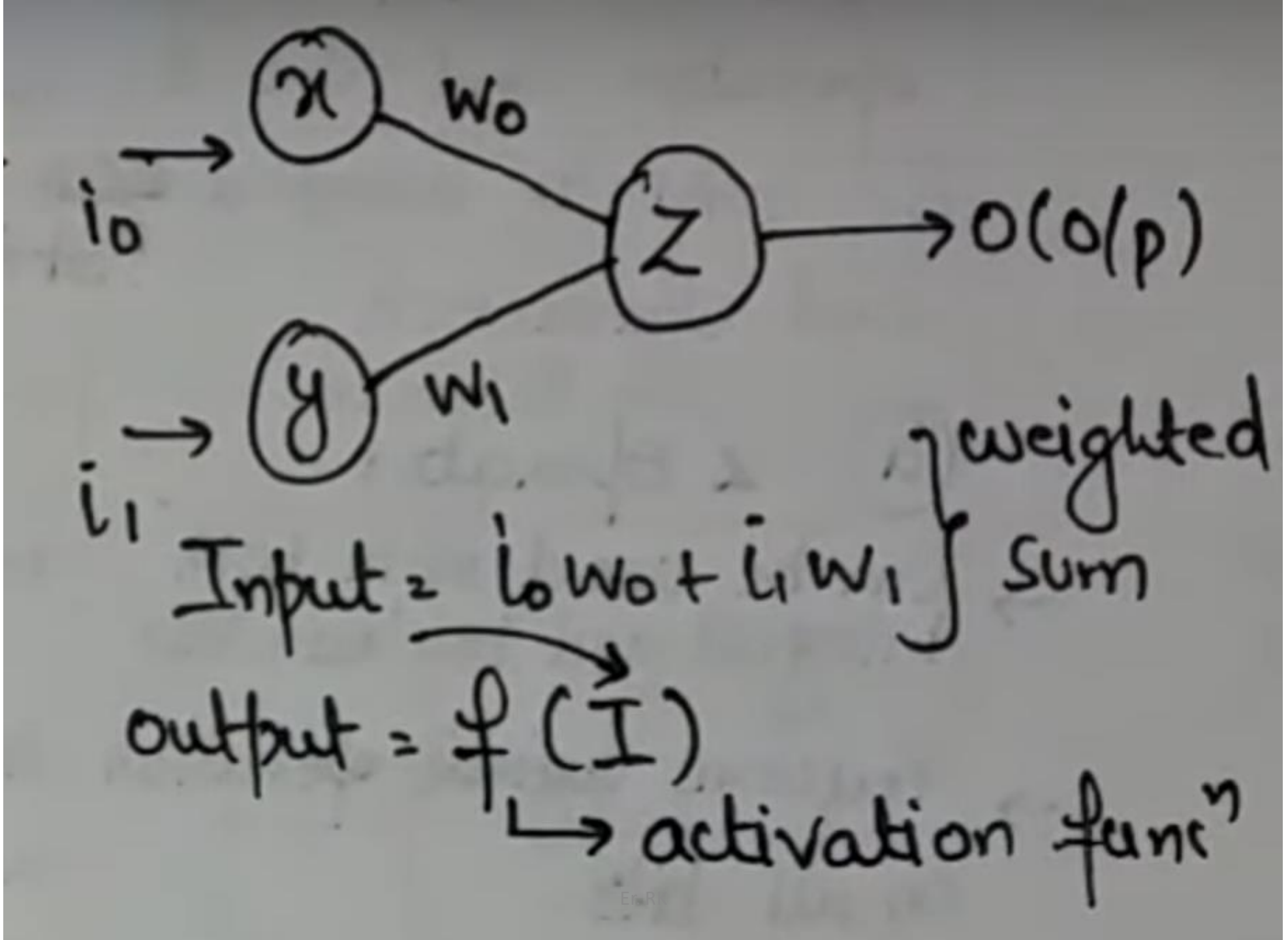
Architecture

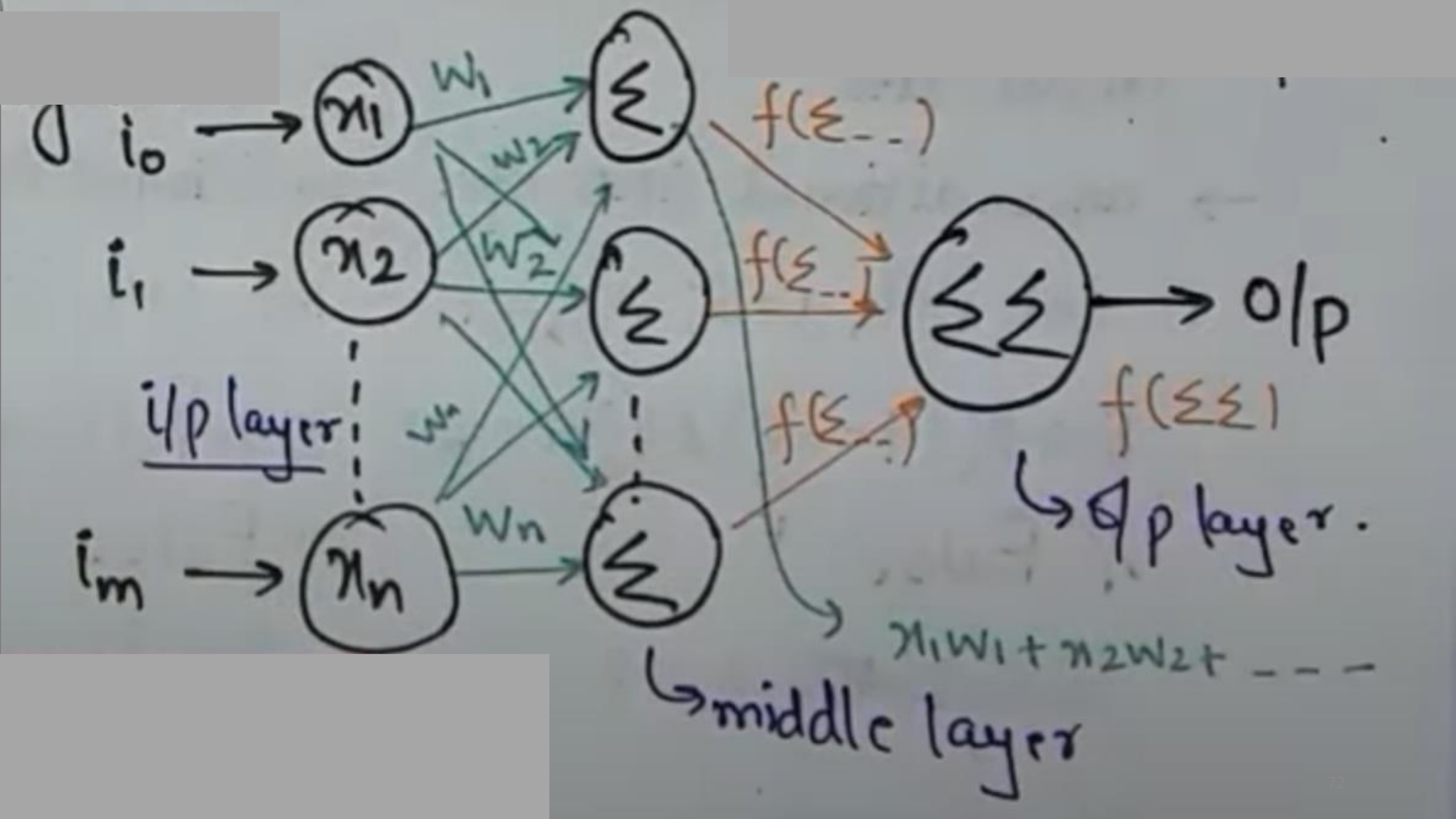


$$y = f(y_{in})$$

$$y_{in} = x_1 w_1 + x_2 w_2$$

Each layer can have multiple neurons, and layers can vary in number.


x
w0
i0
z
o(o/p)
y
w1
i1
Input = i0 w0 + i1 w1
weighted sum
output = f(I)
activation func^n


i_0
i_1
i_m
x_1
x_2
x_n
w_1
w_2
w_n
Σ
$f(\Sigma\_\_)$
$f(\Sigma\_\_)$
$f(\Sigma\_\_)$
$\Sigma\Sigma$
o/p
$f(\Sigma\Sigma)$
i/p layer
middle layer
o/p layer.
$x_1w_1 + x_2w_2 + \_\_\_$

Characteristics of ANN:

↳ i) Neurally Implemented mathematical model.

ii) Huge no. of interconnected processing elements called neurons for processing.

iii) i/p signals arrive at processing elements through Connⁿ and Connected Weights.

Types of Layers

- **Dense/Fully Connected Layer:** Each neuron connected to all neurons in next layer.
- **Convolutional Layer:** For images — detects patterns.
- **Recurrent Layer:** For sequences — has memory.
- **Dropout Layer:** Reduces overfitting by randomly disabling neurons during training.

Types of Artificial Neural Networks

Type	Description
Feedforward Neural Network (FNN)	Data flows in one direction.
Multi-Layer Perceptron (MLP)	Feedforward with one or more hidden layers.
Convolutional Neural Network (CNN)	For image data — detects spatial hierarchies.
Recurrent Neural Network (RNN)	For sequence data — remembers past info.
Deep Neural Network (DNN)	Many hidden layers — deeper learning.

Applications of Artificial Neural Networks

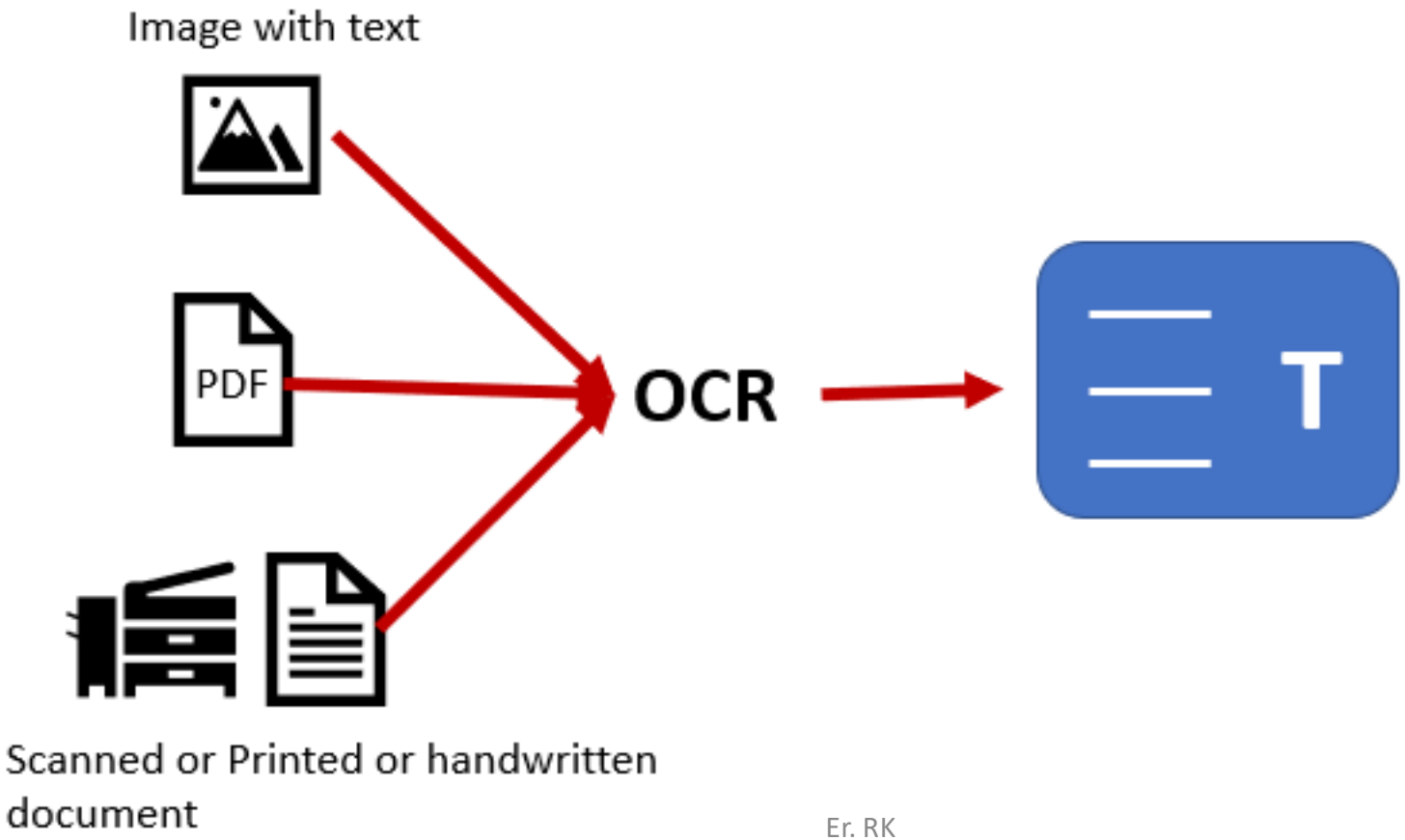
- **Social Media:** Artificial Neural Networks are used heavily in Social Media. For example, let's take the **'People you may know'** feature on Facebook that suggests people that you might know in real life so that you can send them friend requests.
- **Marketing and Sales:** When you log onto E-commerce sites like Amazon and Flipkart, they will recommend your products to buy based on your previous browsing history.

Applications of Artificial Neural Networks

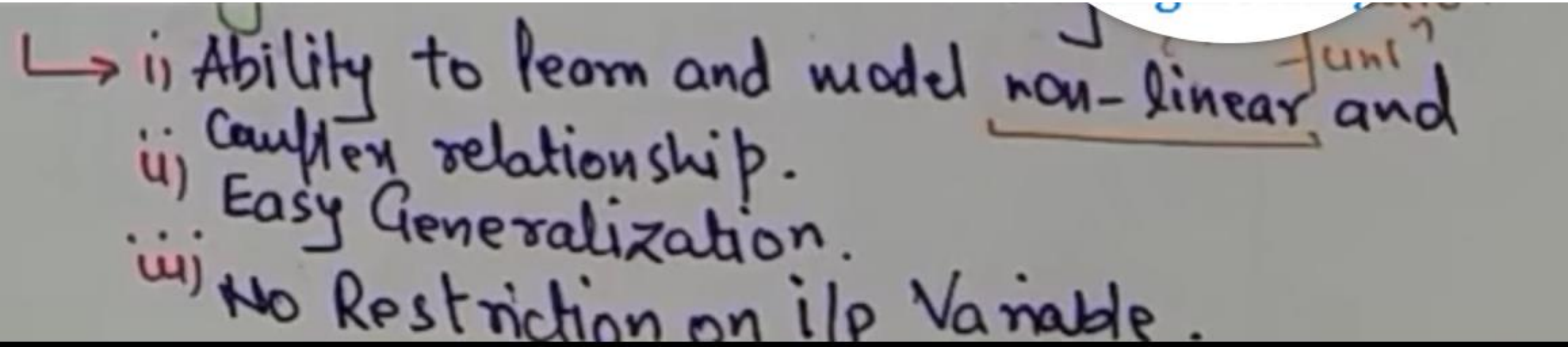
- **Healthcare**: Artificial Neural Networks are used in Oncology to train algorithms that can identify cancerous tissue at the microscopic level at the same accuracy as trained physicians.
- **Personal Assistants:** I am sure you all have heard of Siri, Alexa, Cortana, etc., and also heard them based on the phones you have!!! These are personal assistants and an example of speech recognition that uses **Natural Language Processing** to interact with the users and formulate a response accordingly.

Applications of Artificial Neural Networks

- **Image Processing and Character recognition**
-



Advantages

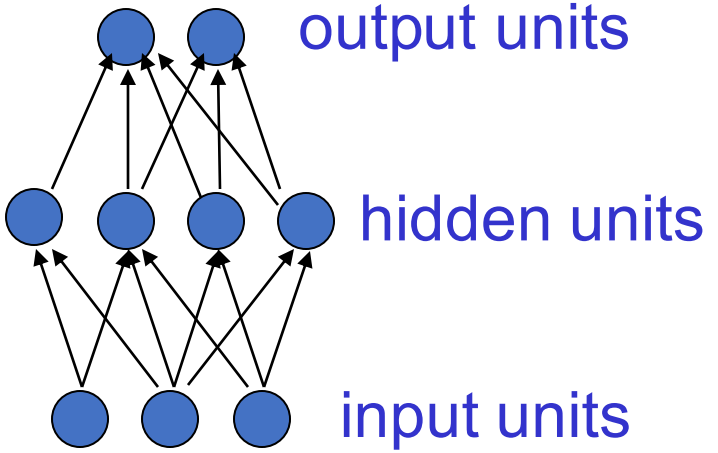

i) Ability to learn and model non-linear and
complex relationship.
ii) Easy Generalization.
iii) No Restriction on i/p Variable.

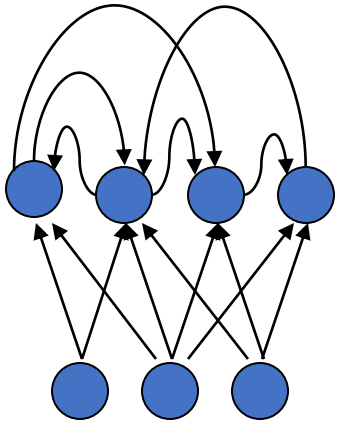
Disadvantages of Artificial Neural Networks

- **Hardware Dependence**
- **Understanding the network's operation**
- **Assured network structure:**
- **Difficulty in presenting the issue to the network**
- **The network's lifetime is unknown**

Types of connectivity

- Feedforward networks
  - These compute a series of transformations
  - Typically, the first layer is the input and the last layer is the output.
- Recurrent networks
  - These have directed cycles in their connection graph. They can have complicated dynamics.
  - More biologically realistic.





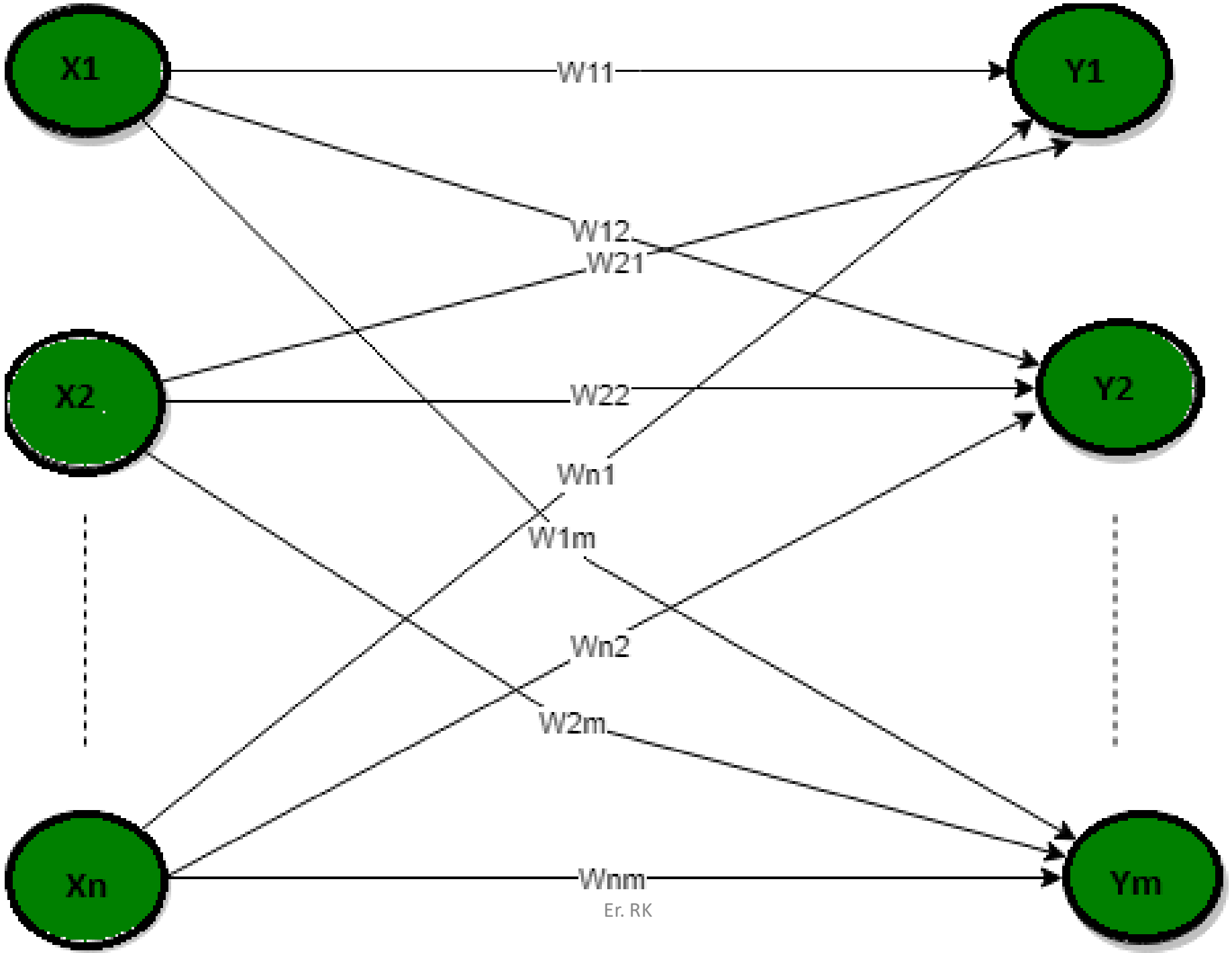
Types of ANN

- There exist five basic types of neuron connection architecture :

1. Single-layer feed-forward network
2. Multilayer feed-forward network
3. Single node with its own feedback
4. Single-layer recurrent network
5. Multilayer recurrent network

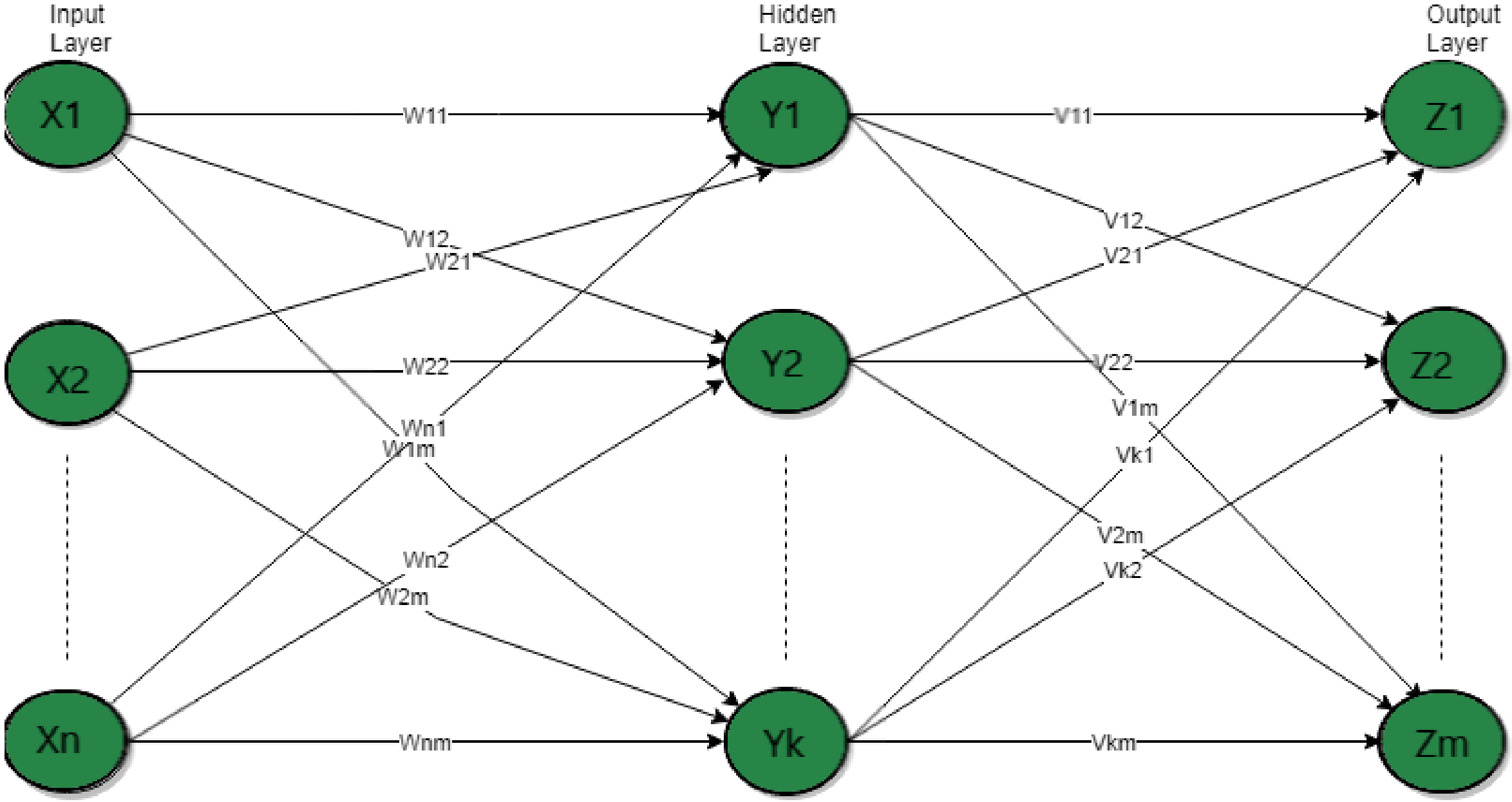
Single-layer feed-forward network

- We have only two layers input layer and the output layer but the input layer does not count because no computation is performed in this layer.
- The output layer is formed when different weights are applied to input nodes and the cumulative effect per node is taken.
- After this, the neurons collectively give the output layer to compute the output signals.


X1
W11
Y1
W12
W21
X2
W22
Y2
Wn1
W1m
Wn2
W2m
Xn
Wnm
Ym

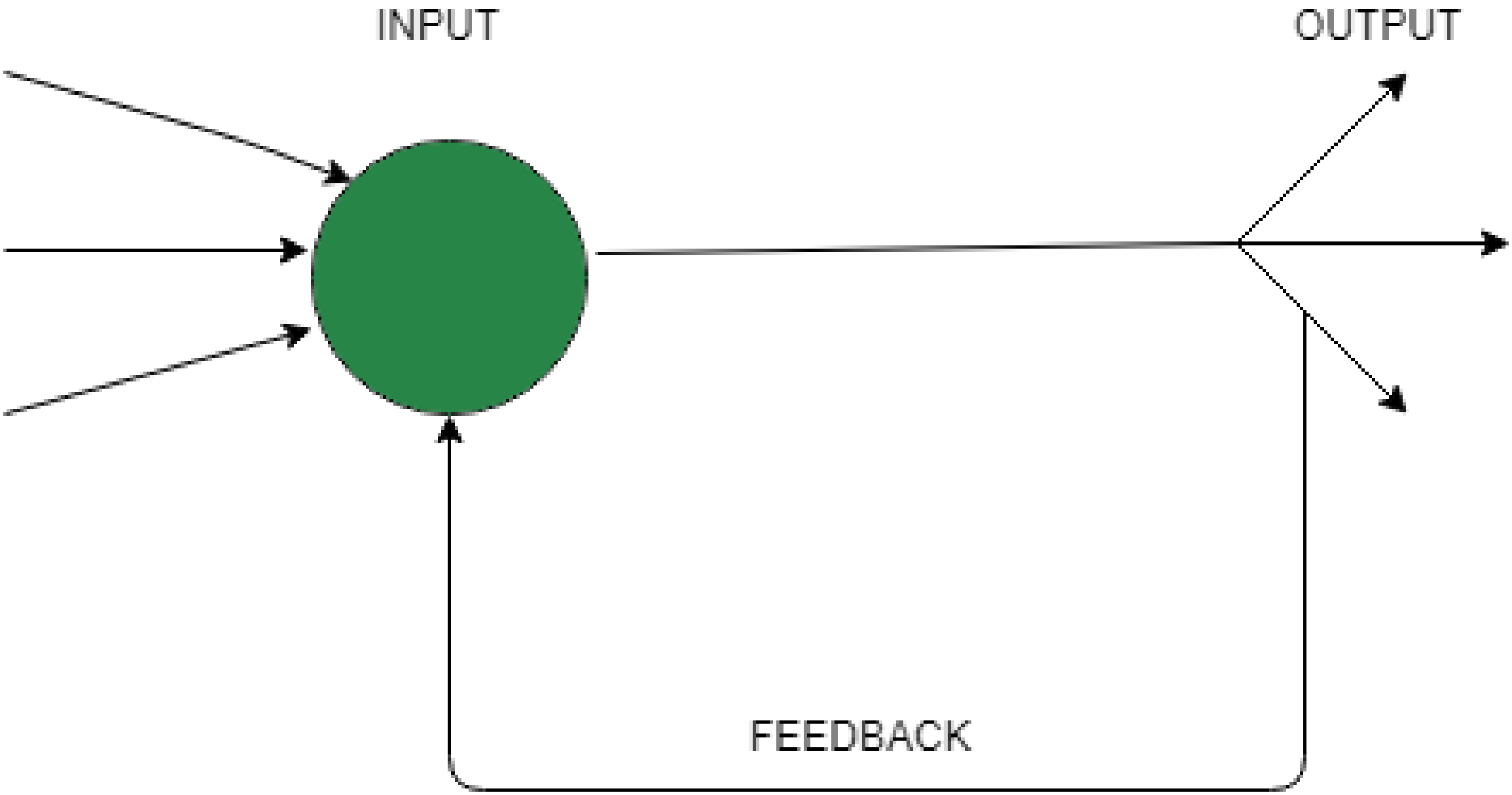
Multilayer feed-forward network

- This layer also has a hidden layer that is internal to the network and has no direct contact with the external layer.
- The existence of one or more hidden layers enables the network to be computationally stronger, a feed-forward network because of information flow through the input function, and the intermediate computations used to determine the output Z.
- There are no feedback connections in which outputs of the model are fed back into itself.


Input Layer
Hidden Layer
Output Layer
X1
X2
Xn
Y1
Y2
Yk
Z1
Z2
Zm
W11
W12
W21
W22
Wn1
W1m
Wn2
W2m
Wnm
V11
V12
V21
V22
V1m
Vk1
V2m
Vk2
Vkm

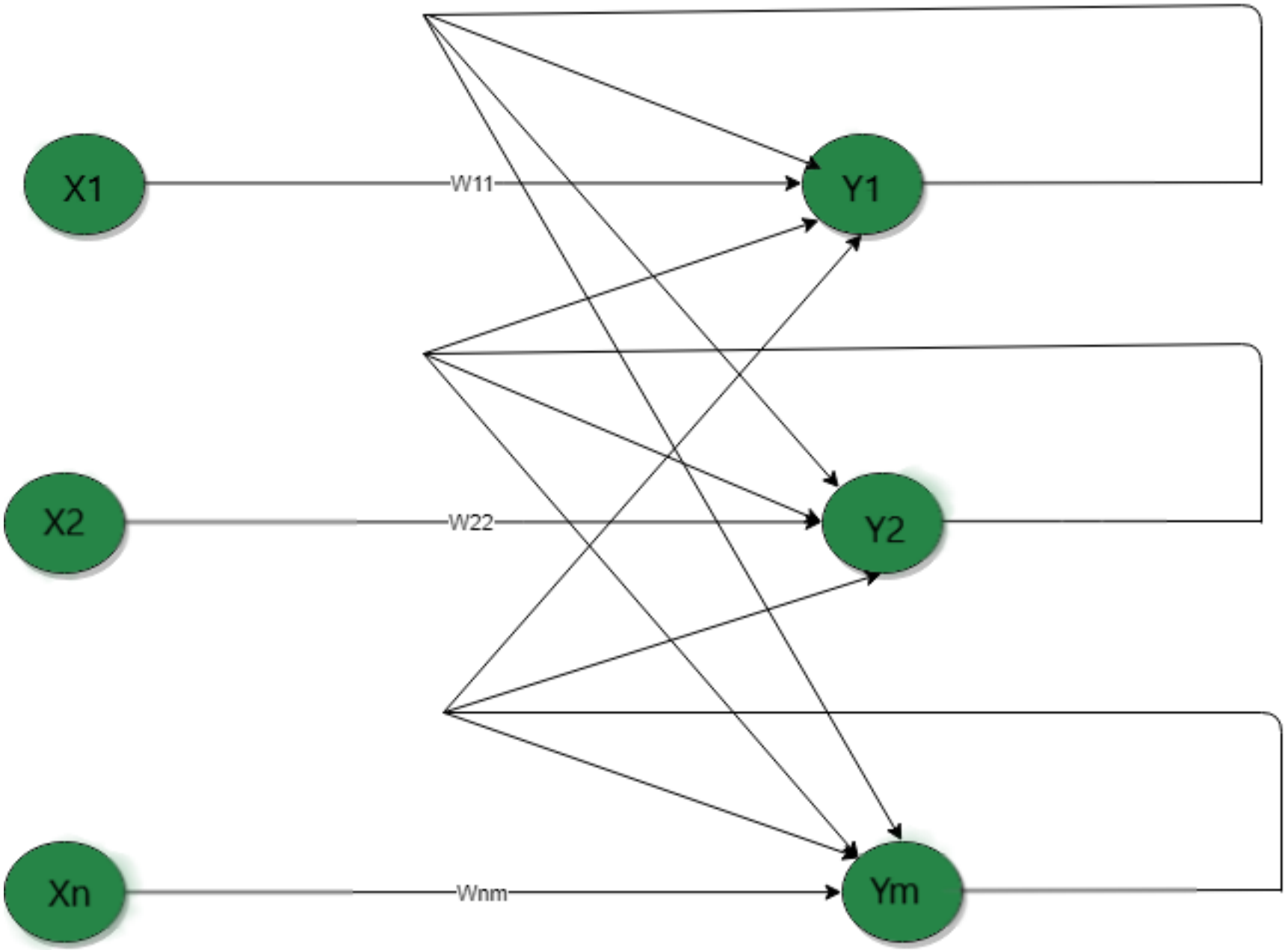
Single node with its own feedback

- When outputs can be directed back as inputs to the same layer or preceding layer nodes, then it results in feedback networks.
- Recurrent networks are feedback networks with closed loops.
- The figure shows a single recurrent network having a single neuron with feedback to itself.


INPUT
OUTPUT
FEEDBACK

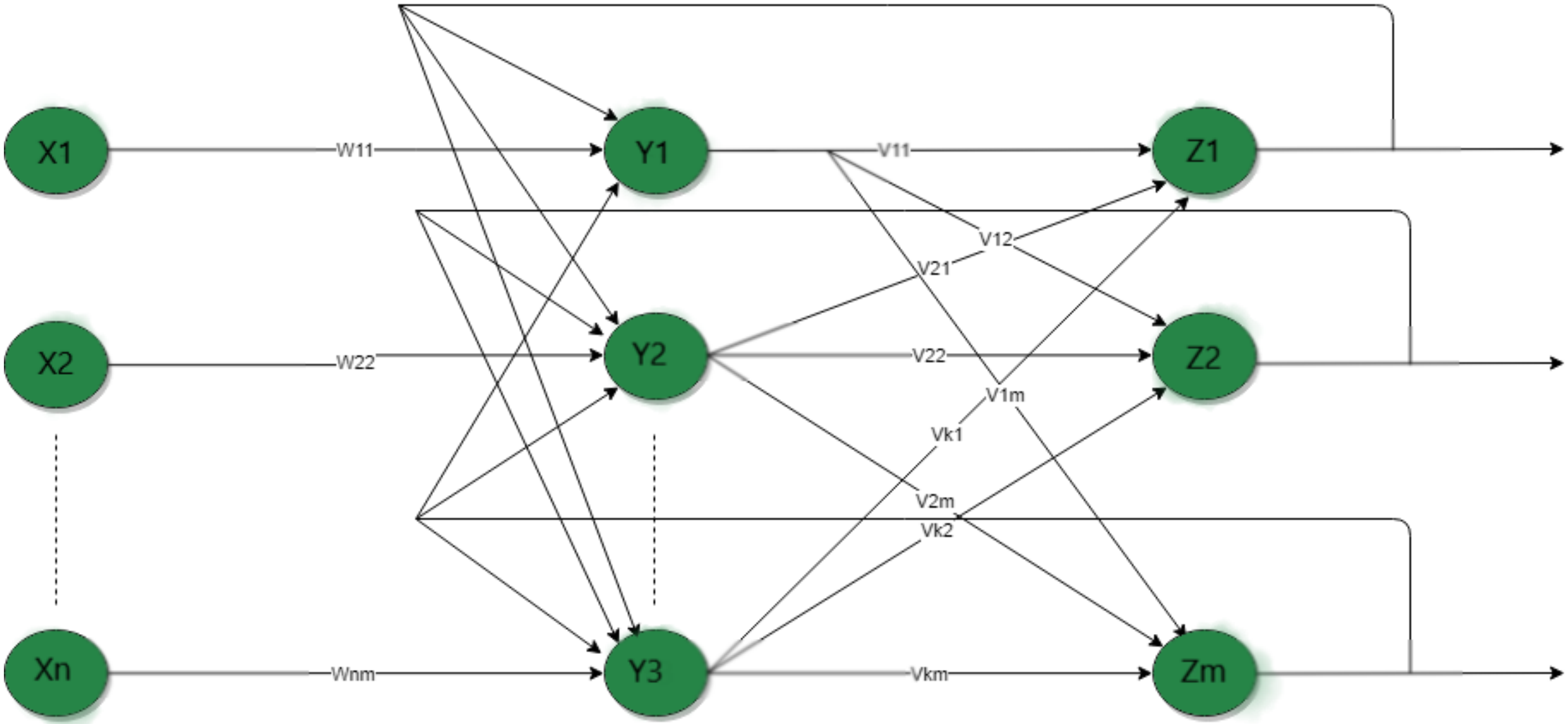
Single-layer recurrent network

- The above network is a single-layer network with a feedback connection in which the processing element's output can be directed back to itself or to another processing element or both.
- A recurrent neural network is a class of artificial neural networks where connections between nodes form a directed graph along a sequence.
- This allows it to exhibit dynamic temporal behavior for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs.


X1
W11
Y1
X2
W22
Y2
Xn
Wnm
Ym

Multilayer recurrent network

- In this type of network, processing element output can be directed to the processing element in the same layer and in the preceding layer forming a multilayer recurrent network.
- They perform the same task for every element of a sequence, with the output being dependent on the previous computations. Inputs are not needed at each time step.
- The main feature of a Recurrent Neural Network is its hidden state, which captures some information about a sequence.


X1
X2
Xn
W11
W22
Wnm
Y1
Y2
Y3
V11
V12
V21
V22
V1m
Vk1
V2m
Vk2
Vkm
Z1
Z2
Zm

Comparison between brain verses computer

	Brain	ANN
Speed	Few ms.	Few nano sec. massive ||el processing
Size and complexity	10^{11} neurons & 10^{15} interconnections	Depends on designer
Storage capacity	Stores information in its interconnection or in synapse. No Loss of memory	Contiguous memory locations loss of memory may happen sometimes.
Tolerance	Has fault tolerance	No fault tolerance Inf gets disrupted when interconnections are disconnected
Control mechanism	Complicated involves chemicals in biological neuron	Simpler in ANN

Activation Function

In an ANN, each neuron computes:

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

Here, z is just a weighted sum of inputs + bias — a **linear value**.

But real-world problems are **non-linear** (like language, vision, etc.),

so we apply an **activation function** $f(z)$ to add **non-linearity** into the model.

Why Do We Need Activation Functions?

- Without an activation function, a neural network (even with multiple layers) is just a linear model: $y=Wx+b$
- Adding activation functions allows:
  - the network to model **complex, non-linear patterns**,
  - learn better decision boundaries,
  - and approximate any function.

Key Roles of Activation Functions:

Role	Explanation
Non-Linearity	Allows the network to learn non-linear relationships
Decision Boundaries	Helps distinguish between classes or outputs
Stacking Layers	Makes each layer more expressive & powerful
Normalization	Some functions (like sigmoid) squash outputs into a fixed range

Types of Activation Functions:

🔷 1️⃣ Linear Activation:

$$f(z) = z$$

✅ Rarely used now — as it does **not introduce non-linearity**.

✅ Issue: no matter how many layers you stack → still linear.

🔷 2️⃣ Step Function:

$$f(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

✅ Used in **perceptrons** (early models).

✅ Output: binary (0 or 1).

✅ Problem: not differentiable → can't use gradient descent.

🔷 3️⃣ Sigmoid:

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

✅ Output: between 0 and 1 → good for probabilities.

✅ Smooth, differentiable.

✅ Problems:

- Vanishing gradient for large $|z|$ (gradients become very small → slow learning).

🔷 4️⃣ Tanh (Hyperbolic Tangent):

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

✅ Output: between -1 and 1.

✅ Zero-centered → better than sigmoid.

✅ Still suffers from vanishing gradients at extremes.

🔷 5️⃣ ReLU (Rectified Linear Unit):

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

✅ Most widely used.

✅ Output: 0 if $z < 0$; z if $z > 0$.

✅ Advantages:

- Simple, fast, and efficient.
- No vanishing gradients for $z > 0$.

✅ Problem: "Dead ReLU" → neurons can stop updating if $z < 0$ too often.

🔷 6️⃣ Leaky ReLU:

$$f(z) = \begin{cases} z, & z \geq 0 \\ 0.01z, & z < 0 \end{cases}$$

✅ Fixes “Dead ReLU” by allowing a small slope when $z < 0$.

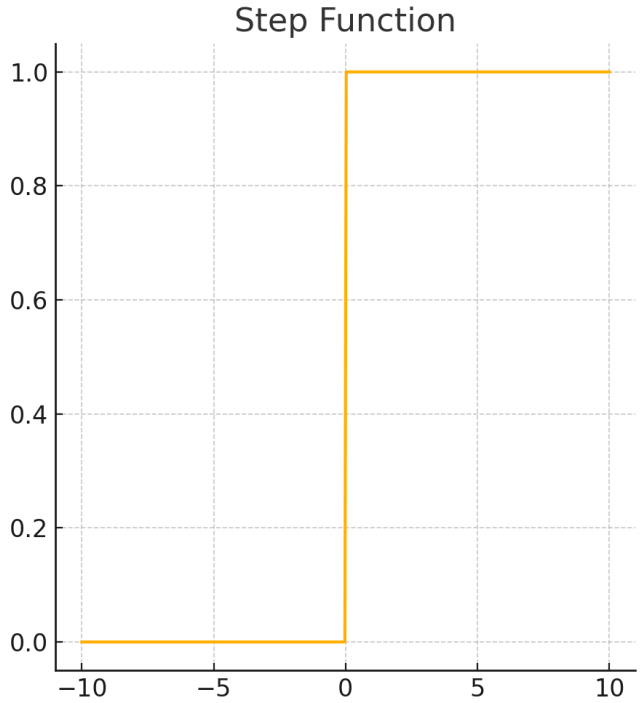
🔷 7️⃣ Softmax:

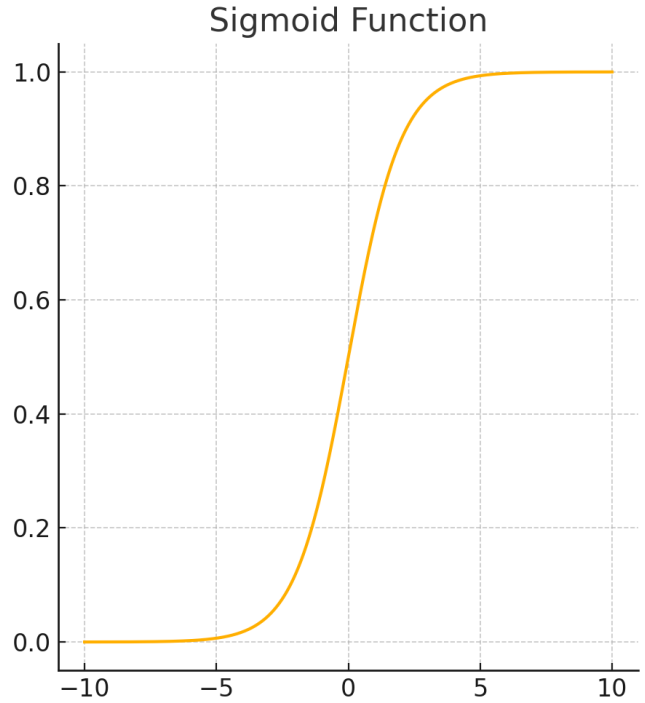
For multi-class classification:

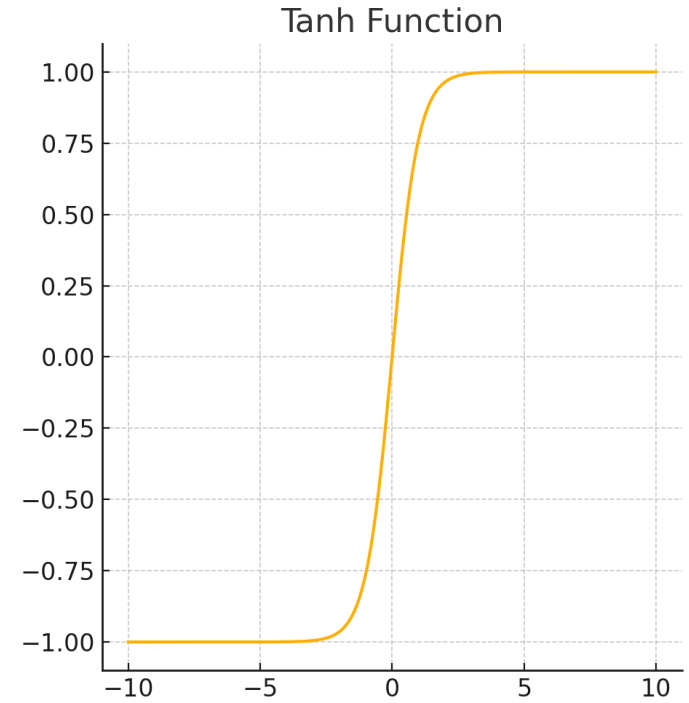
$$f(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

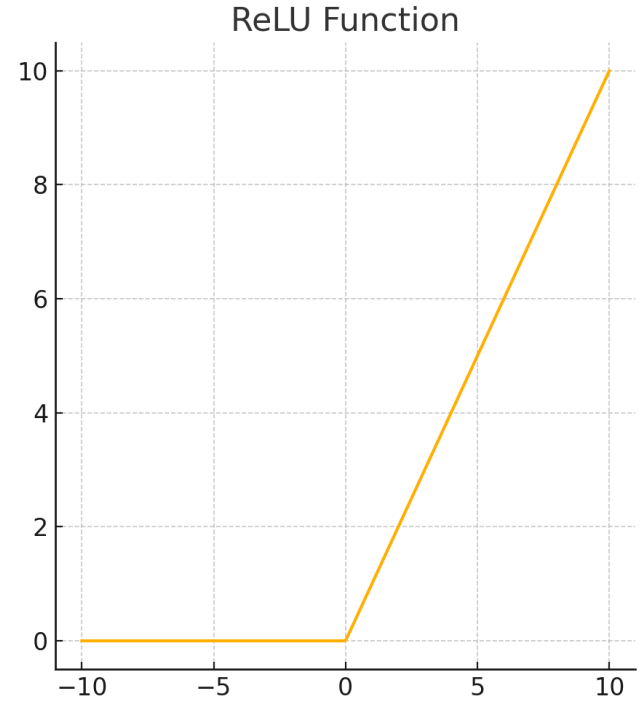
✅ Outputs probabilities for each class.

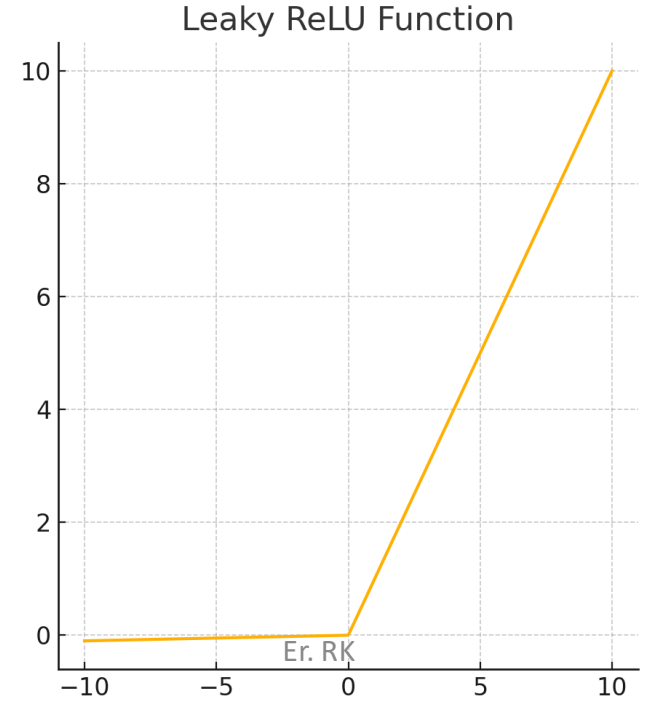
✅ Used at the **output layer** for classification problems.

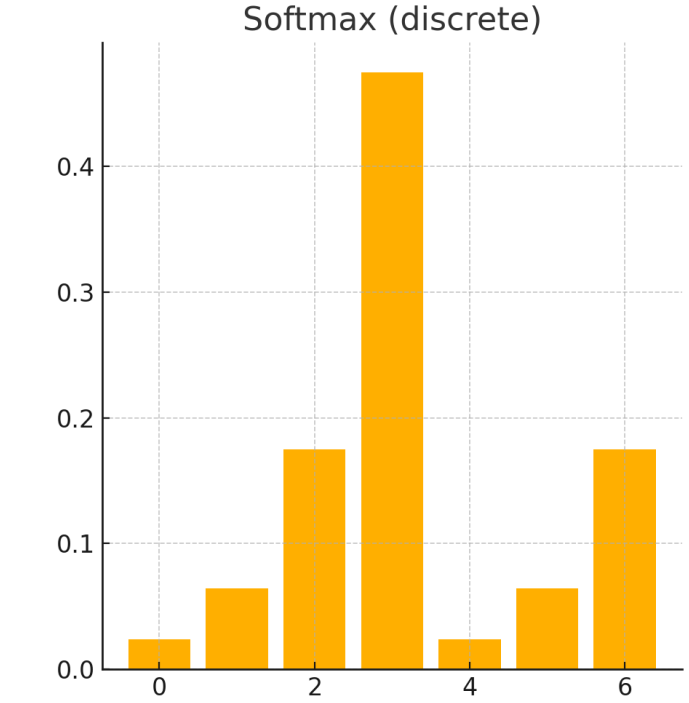

Step Function
1.0
0.8
0.6
0.4
0.2
0.0
−10
−5
0
5
10


Sigmoid Function
1.0
0.8
0.6
0.4
0.2
0.0
−10
−5
0
5
10


Tanh Function
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10
−5
0
5
10


ReLU Function
10
8
6
4
2
0
−10
−5
0
5
10


Leaky ReLU Function
10
8
6
4
2
0
−10
−5
0
5
10


Softmax (discrete)
0.4
0.3
0.2
0.1
0.0
0
2
4
6

Without Activation:

$$y = w_1x_1 + w_2x_2 + b$$

If y is always linear → can't model curves or complex boundaries.

With Activation (e.g., ReLU):

$$y = \text{ReLU}(w_1x_1 + w_2x_2 + b)$$

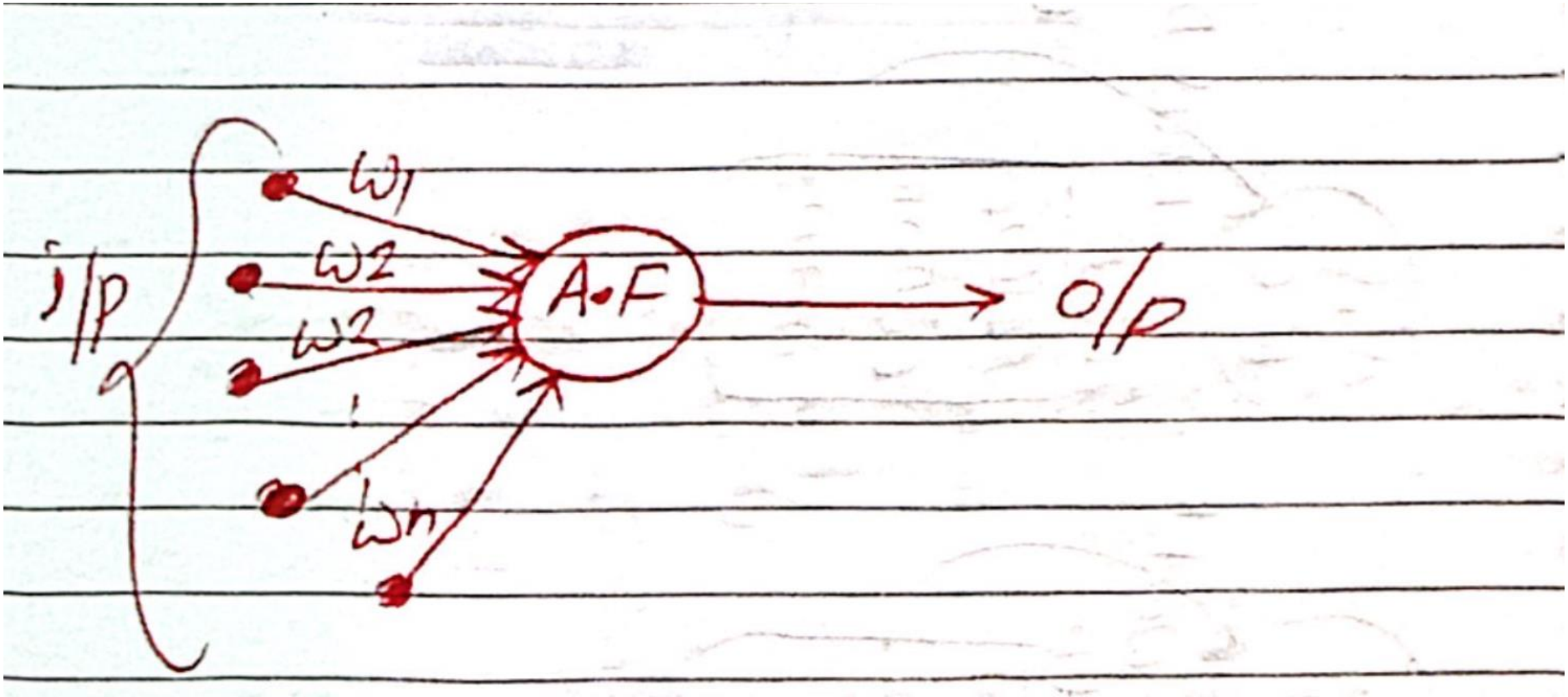
Now, y can capture non-linear behavior and learn more complex patterns.

Activation function:-

→ It is internal state of neuron

→ used to convert the i/p signal of node of ANN to an o/p signal

→ weighted sum of i/p becomes i/p signal to A.F to give one o/p signal


I/P
w1
w2
w2
wn
A.F
O/P

Importance

→ They introduce non-linear properties to n/w.

Without Activation Function:

- o/p signal would be simple linear function. i.e easy to solve but limited to processing complex learning.

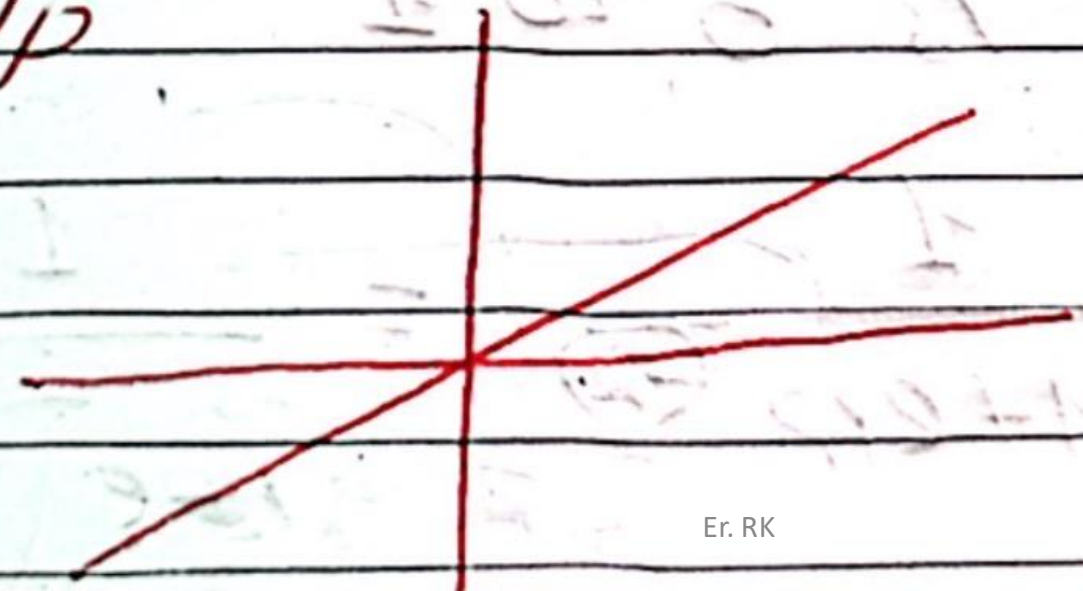
simple linear function means polynomial of one degree.

{ Limited to images, video, audio }

Types

(i) Identity function :- $f(x) = x$ for all x

→ linear function, dp is same as i/p



(ii) Binary step function

→ single layer n/w to convert net i/p to o/p

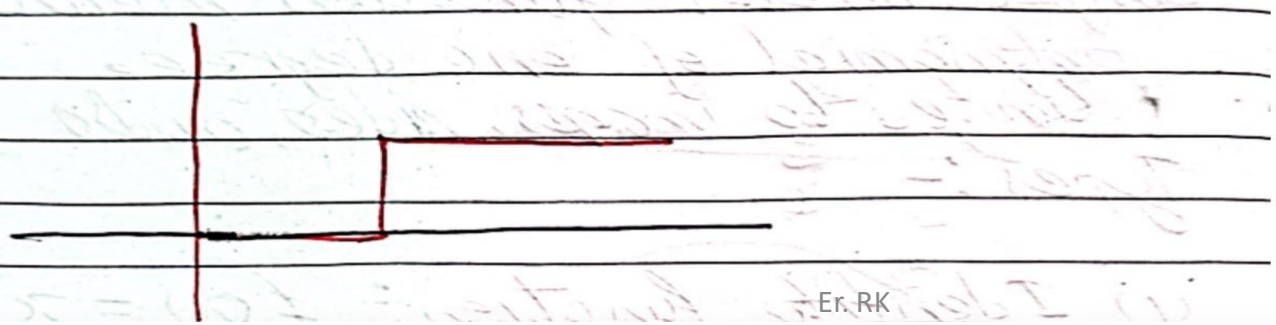
i/p do

o/p → 0

o/p → 1

$f(x) = 1$ if $x >= t$ } i.e t = threshold

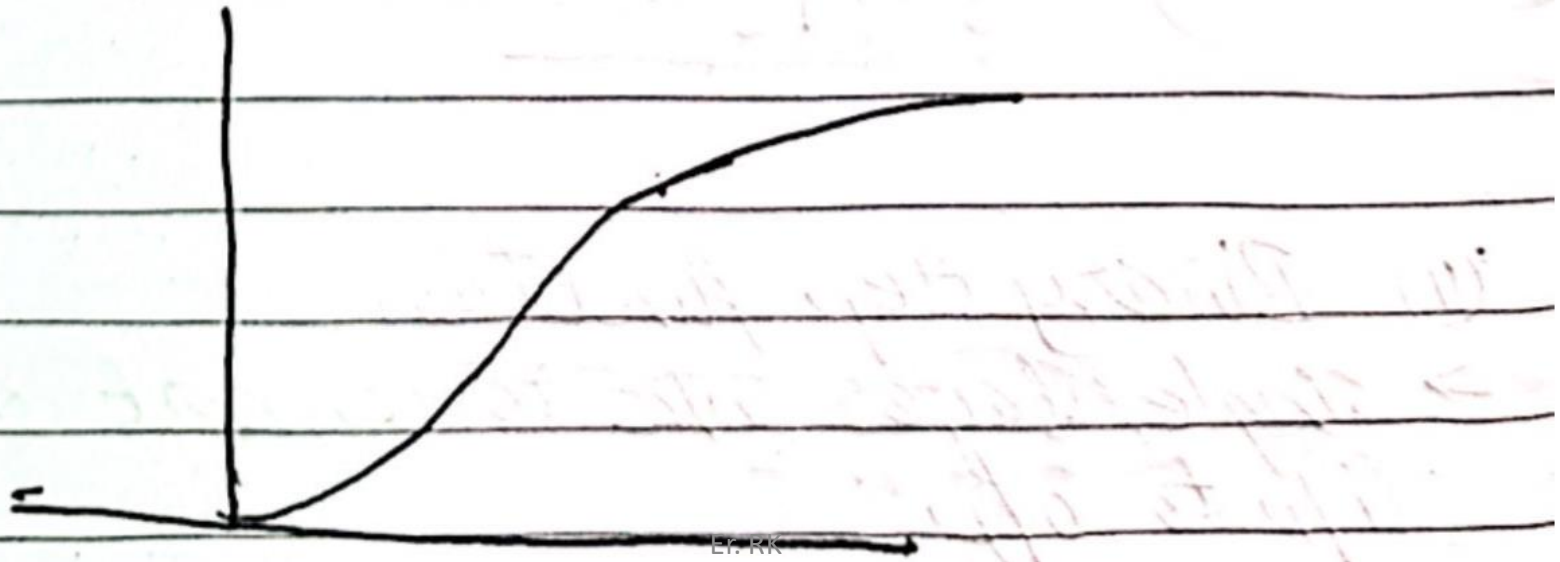
0 if $x < t$ }



(iii) Sigmoidal function:

- used in back propagation n/w
- range is 0 to 1

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

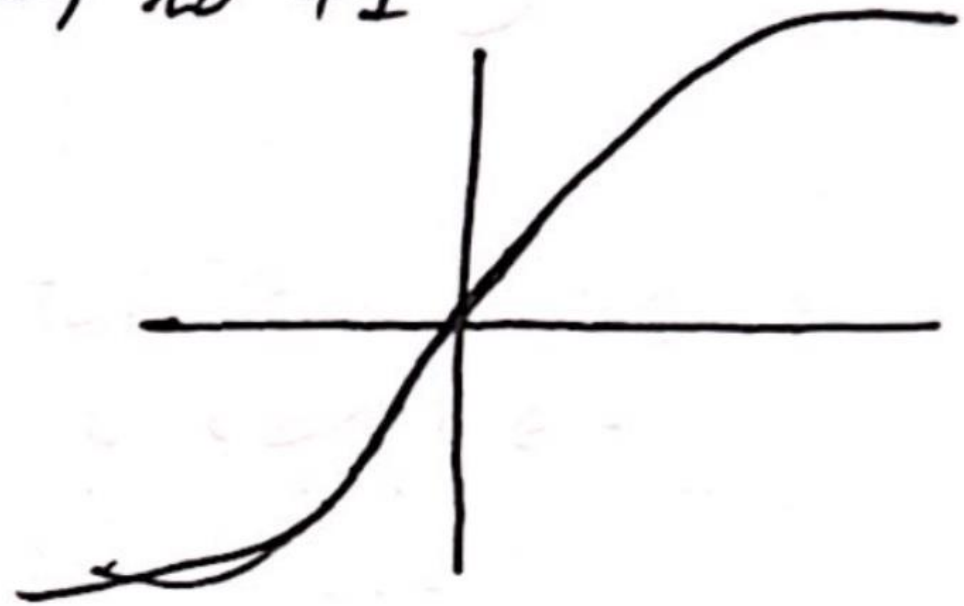


(iv) HyperLolic Tangent function (Tanh) :-

→ Easy optimization is easy

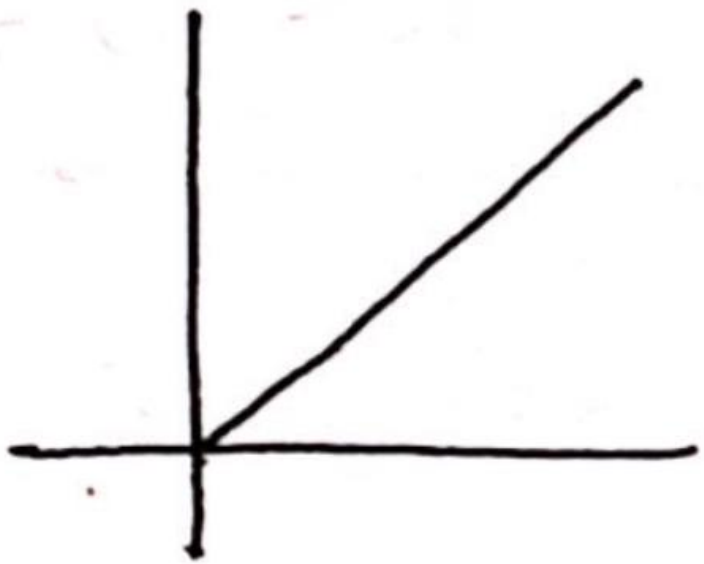
→ Nw Range is −1 to +1

$$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$$



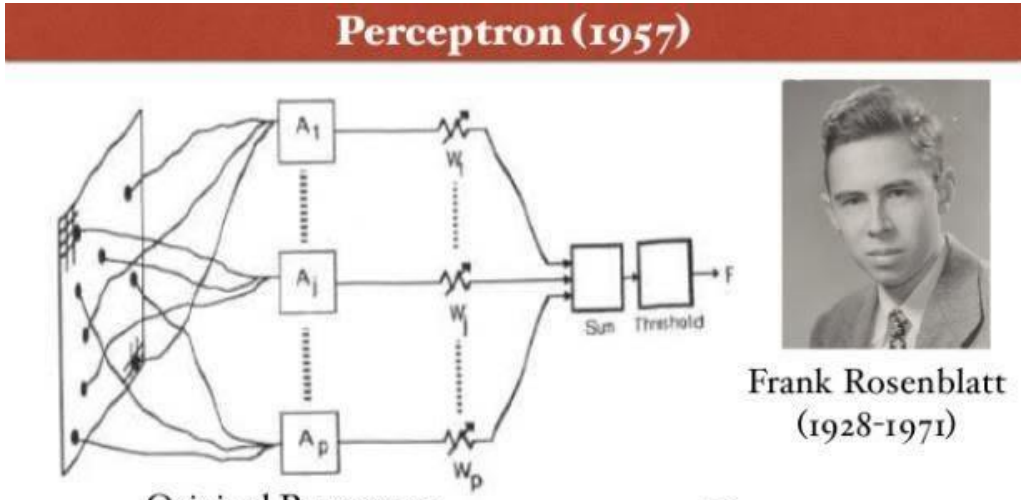
(v) Ramp function

$$R(x) = \begin{cases} x, & x \geq 0 \\ 0 & x < 0 \end{cases}$$



Perceptron

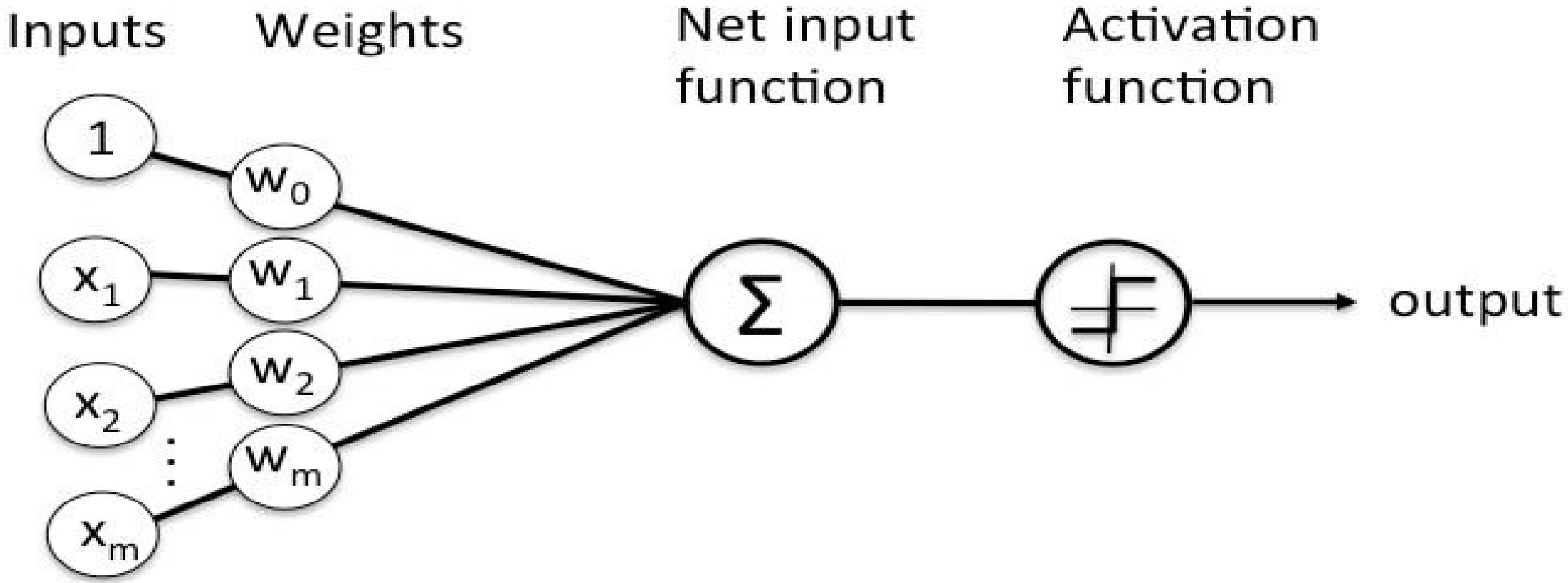
- A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data.



Perceptron

- Perceptron was introduced by Frank Rosenblatt in 1957.
- He proposed a Perceptron learning rule based on the original MCP neuron.
- A Perceptron is an algorithm for supervised learning of binary classifiers.
- This algorithm enables neurons to learn and processes elements in the training set one at a time.

Perceptron

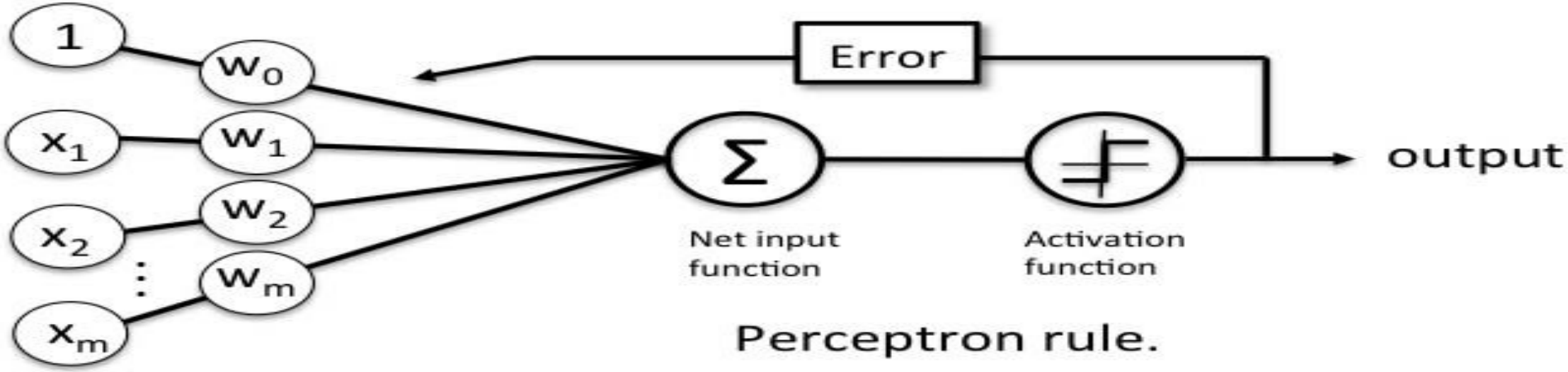

Inputs
Weights
Net input function
Activation function
1
x_1
x_2
x_m
w_0
w_1
w_2
w_m
Σ
output

Perceptron

- There are two types of Perceptrons: Single layer and Multilayer.
- Single layer Perceptrons can learn only linearly separable patterns.
- Multilayer Perceptrons or feedforward neural networks with two or more layers have the greater processing power.
- The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary.
- This enables you to distinguish between the two linearly separable classes +1 and -1.
- Note: Supervised Learning is a type of Machine Learning used to learn models from labeled training data. It enables output prediction for future or unseen data.

Perceptron Learning Rule

- Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients.
- The input features are then multiplied with these weights to determine if a neuron fires or not.



Perceptron rule.

Perceptron function

- Perceptron is a function that maps its input “x,” which is multiplied with the learned weight coefficient; an output value ”f(x)”is generated.

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

- In the equation given above:

  “w” = vector of real-valued weights

  “b” = bias (an element that adjusts the boundary away from origin without any dependence on the input value)

  “x” = vector of input x values

Perceptron function

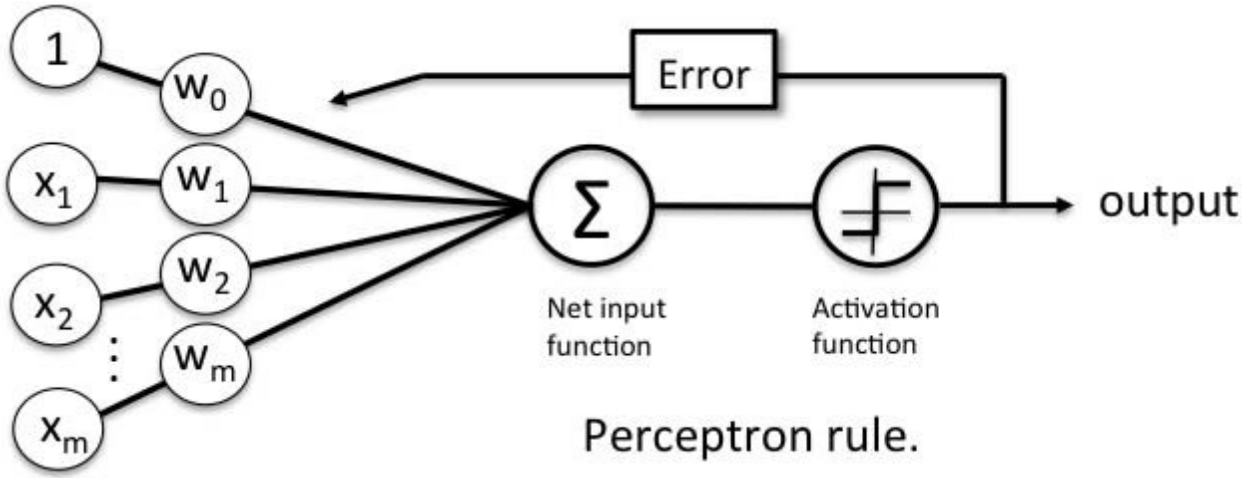
"m" = number of inputs to the Perceptron

$$\sum_{i=1}^{m} w_i x_i$$

- The output can be represented as "1" or "0."It can also be represented as "1" or "-1" depending on which activation function is used.

Inputs of Perceptron

- A Perceptron accepts inputs, moderates them with certain weight values, then applies the transformation function to output the final result. The above below shows a Perceptron with a Boolean output.



Perceptron rule.

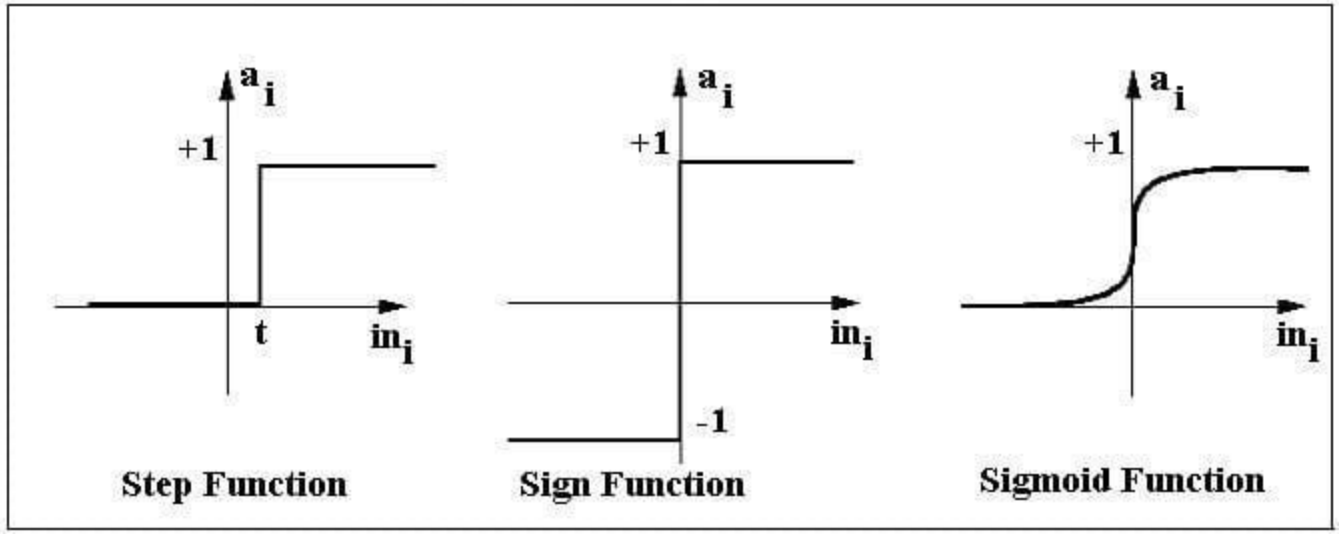
Inputs of Perceptron

- • A Boolean output is based on inputs such as salaried, married, age, past credit profile, etc. It has only two values: Yes and No or True and False.
- • The summation function "∑" multiplies all inputs of "x" by weight "w" and then adds them up as follows:

$$w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

Activation function

- The activation function applies a step rule (convert the numerical output into +1 or -1) to check if the output of the weighting func is greater than zero or not.



Example

- For example:

  If $\sum wixi > 0$ => then final output “o” = 1 (issue bank loan)

  Else, final output “o” = -1 (deny bank loan)

- Step function gets triggered above a certain value of the neuron output; else it outputs zero. Sign Function outputs +1 or -1 depending on whether neuron output is greater than zero or not. Sigmoid is the S-curve and outputs a value between 0 and 1.

Output of Perceptron

- Perceptron with a Boolean output:

- Inputs: x1…xn

- Output: o(x1….xn)

$$o(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

- Weights: wi=> contribution of input xi to the Perceptron output;

  w0=> bias or threshold

Output of Perceptron

- If $\sum w.x > 0$, output is +1, else -1. The neuron gets triggered only when weighted input reaches a certain threshold value.

$$o(\vec{x}) = sgn(\vec{w} \cdot \vec{x})$$

$$sgn(y) = \begin{cases} 1 \text{ if } y > 0 \\ -1 \text{ otherwise} \end{cases}$$

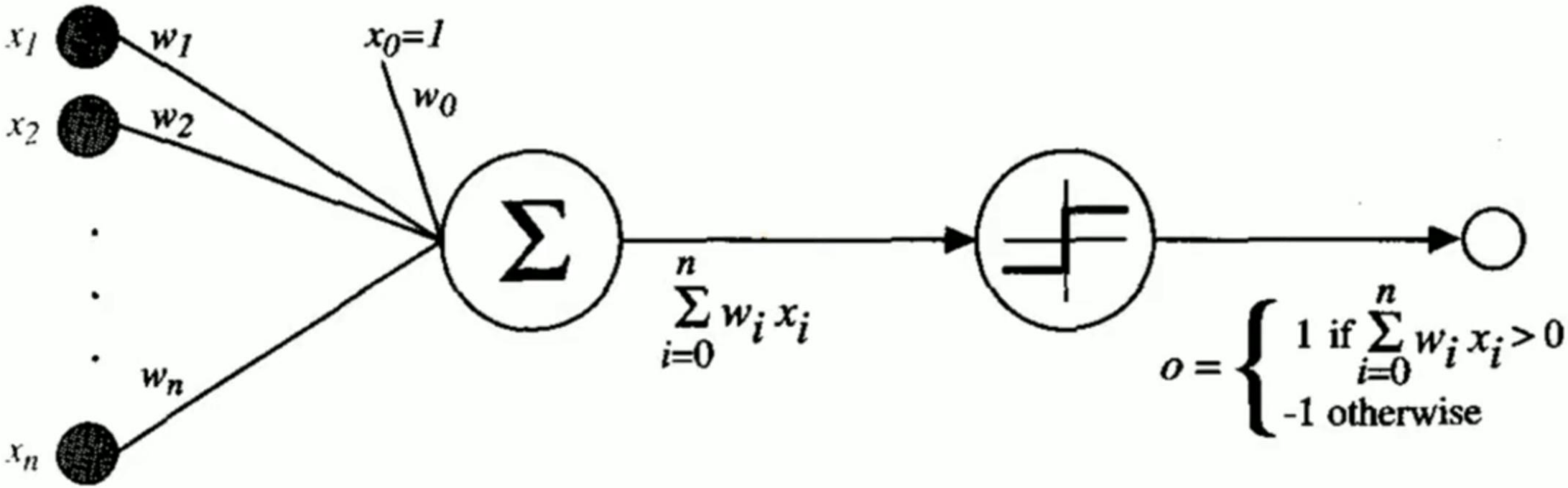
- An output of +1 specifies that the neuron is triggered. An output of -1 specifies that the neuron did not get triggered.
- “sgn” stands for sign function with output +1 or -1.

Error in Perceptron

- In the Perceptron Learning Rule, the predicted output is compared with the known output.
- If it does not match, the error is propagated backward to allow weight adjustment to happen.

PERCEPTRON TRAINING RULE

Artificial Neural Networks



PERCEPTRON – ANN

- A perceptron unit is used to build the ANN system.
- A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise.
- More precisely, given inputs ***x1*** through ***xn,*** the output ***o(x1,*** . . . , ***xn)*** computed by the perceptron is

$$o(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

- where each ***wi*** is a real-valued constant, or weight, that determines the contribution of input ***xi*** to the perceptron output

PERCEPTRON TRAINING RULE – ANN

- One way to learn an acceptable weight vector is to begin with random weights, then iteratively apply the perceptron to each training example, modifying the perceptron weights whenever it misclassifies an example.
- This process is repeated, iterating through the training examples as many times as needed until the perceptron classifies all training examples correctly.
- Weights are modified at each step according to the ***perceptron training rule,*** which revises the weight ***wi*** associated with input ***xi*** according to the rule

$$w_i \leftarrow w_i + \Delta w_i$$

where

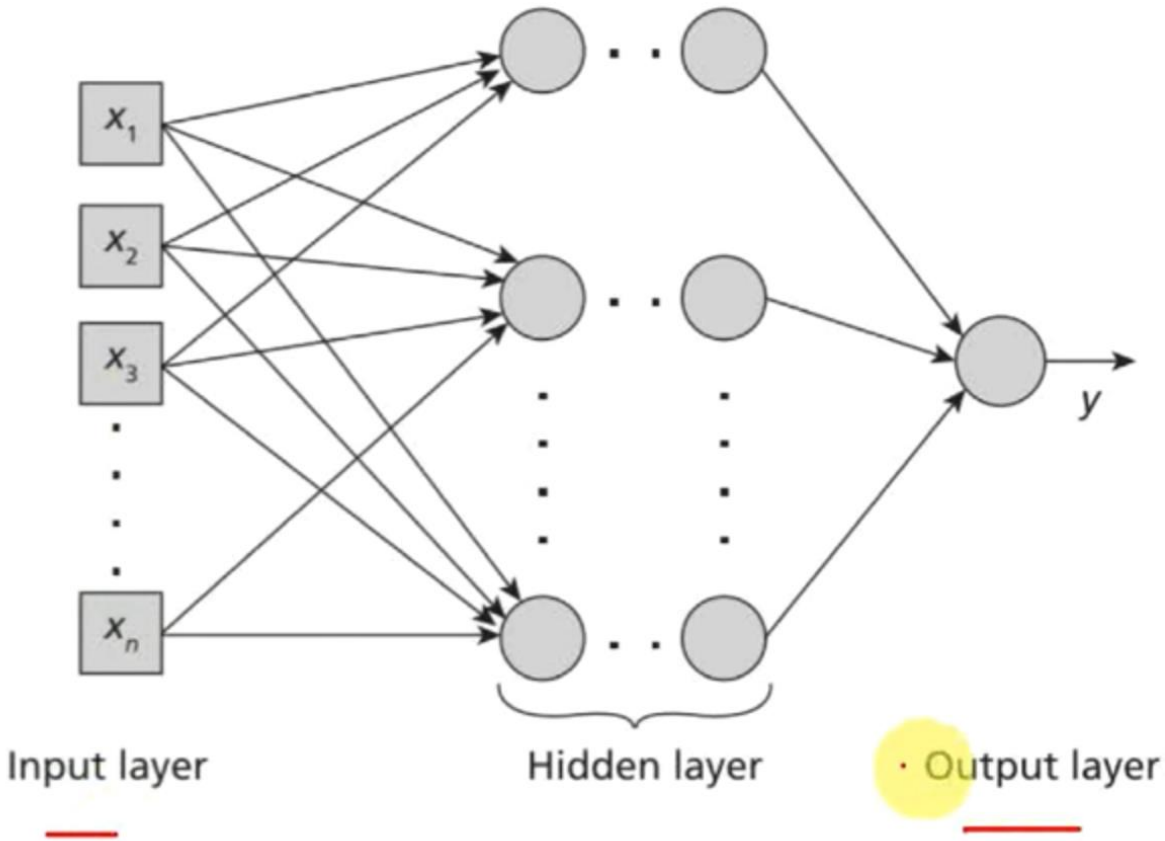
$$\Delta w_i = \eta(t - o)x_i$$

PERCEPTRON TRAINING RULE – ANN

Perceptron_training_rule *(X, η)*
initialize ***w*** *(wi ← an initial (small) random value)*
repeat
    for each training instance $(\mathbf{x}, \mathbf{tx}) \in X$
        compute the real output $\mathbf{ox} = Activation(Summation(\mathbf{w}.\mathbf{x}))$
        if $(\mathbf{tx} \neq \mathbf{ox})$
            for each wi
                $wi \leftarrow wi + \Delta wi$
                $\Delta wi \leftarrow \eta\ (tx - ox)xi$
            end for
        end if
    end for
until all the training instances in X are correctly classified
return ***w***

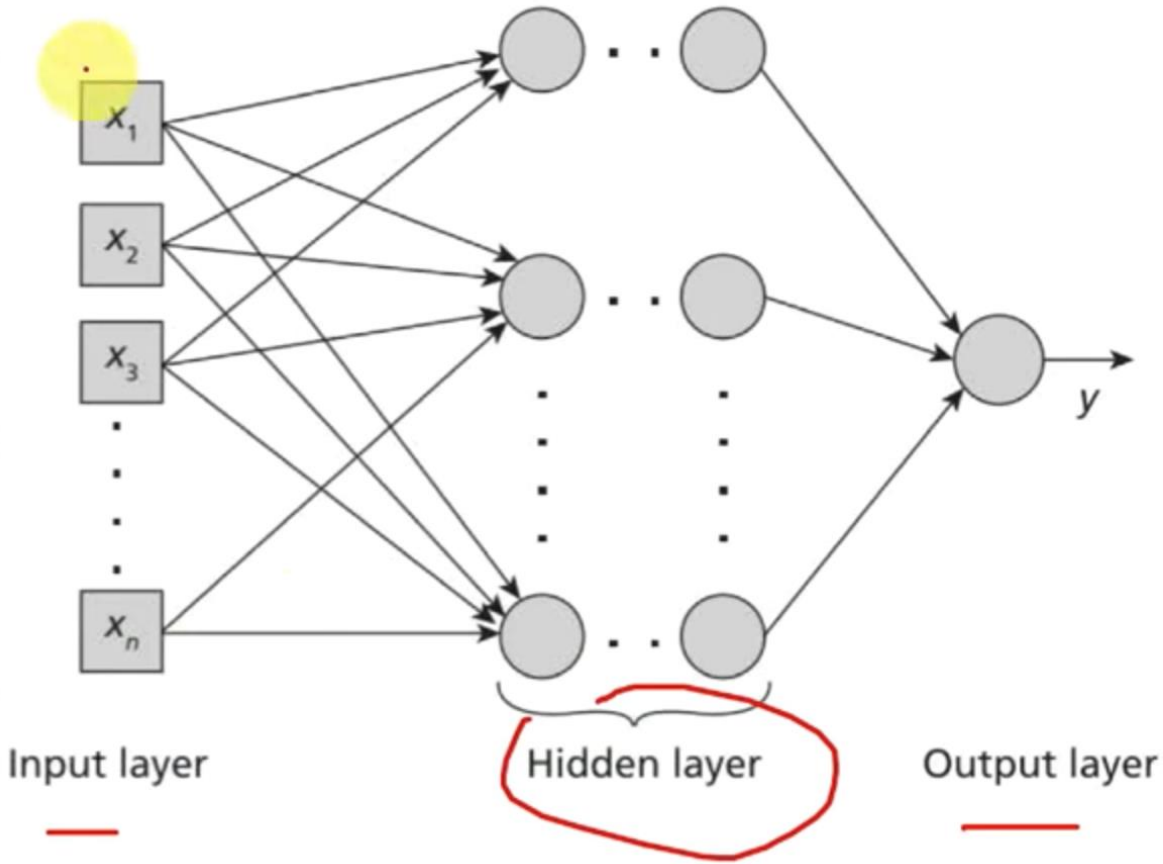
Multi-Layer Perceptron Learning Algorithm

- A multi-layer perceptron is a type of Feed Forward Neural Network with multiple neurons arranged in layers.
- The network has at least three layers with an input layer, one or more hidden layers and an output layer.



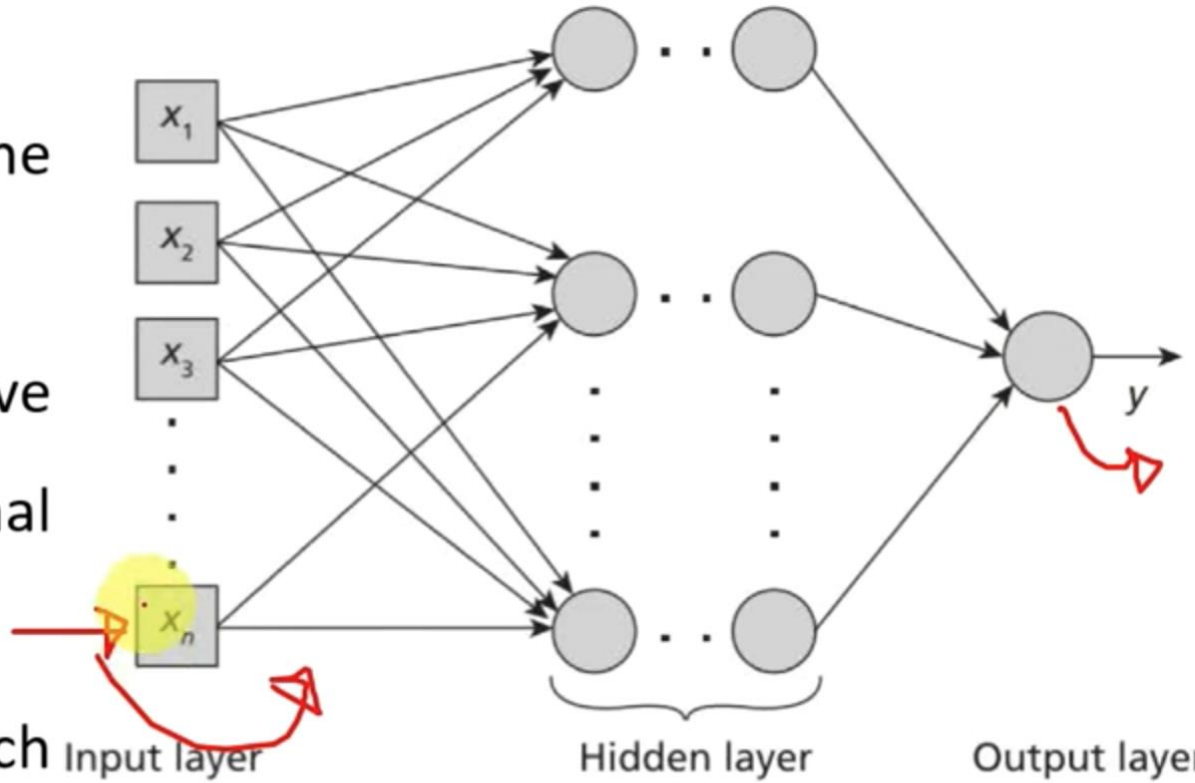
Multi-Layer Perceptron Learning Algorithm

- A multi-layer perceptron is a type of Feed Forward Neural Network with multiple neurons arranged in layers.
- The network has at least three layers with an input layer, one or more hidden layers and an output layer.
- All the neurons in a layer are fully connected to the neurons in the next layer.



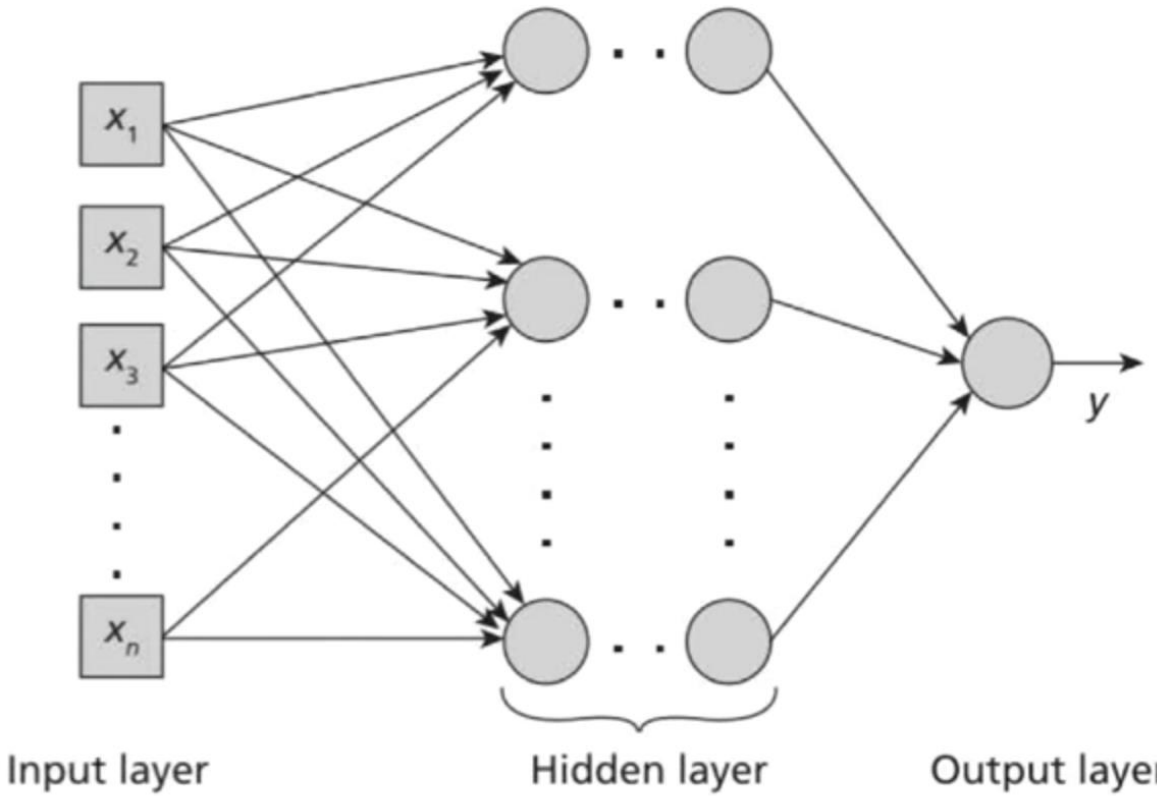
Multi-Layer Perceptron Learning Algorithm

- The input layer is the visible layer.
- It just passes the input to the next layer.
- The layers following the input layer are the hidden layers.
- The hidden layers neither directly receive inputs nor send outputs to the external environment.
- The final layer is the output layer which outputs a single value or a vector of values.



Multi-Layer Perceptron Learning Algorithm

- The activation functions used in the layers can be linear or non-linear depending on the type of the problem modelled.
- Typically, a sigmoid activation function is used if the problem is a binary classification problem and a softmax activation function is used in a multi-class classification problem.



Multi-Layer Perceptron Learning Algorithm

Input: Input vector $(x_1, x_2, ..., x_n)$

Output: Y

Learning rate: α

Assign random weights and biases for every connection in the network in the range [–0.5, +0.5].

Step 1: Forward Propagation

1. **Calculate Input and Output in the *Input Layer*:**

   (Input layer is a direct transfer function, where the output of the node equals the input).

   Input at Node j 'I_j' in the Input Layer is

$$\boxed{I_j = x_j}$$

Multi-Layer Perceptron Learning Algorithm

Input: Input vector $(x_1, x_2, ..., x_n)$

Output: Y_n

Learning rate: α

Assign random weights and biases for every connection in the network in the range [–0.5, +0.5].

Step 1: Forward Propagation

1. **Calculate Input and Output in the *Input Layer*:**

   (Input layer is a direct transfer function, where the output of the node equals the input).

   Input at Node j 'I_j' in the Input Layer is

   $$I_j = x_j$$

   where,

   x_j is the input received at Node j

   Output at Node j 'O_j' in the Input Layer is

   $$O_j = I_j$$

Multi-Layer Perceptron Learning Algorithm

Net Input at Node j in the ***Output Layer*** *is*

$$I_j = \sum_{i=1}^{n} O_i w_{ij} + x_0 \times \theta_j$$

where,

O_i is the output from Node i

w_{ij} is the weight in the link from Node i to Node j

x_0 is the input to bias node '0' which is always assumed as 1

θ_j is the weight in the link from the bias node '0' to Node j

Output at Node j

$$O_j = \frac{1}{1 + e^{-Ij}}$$

where,

I_j is the input received at Node j

Multi-Layer Perceptron Learning Algorithm

3. **Estimate error at the node in the *Output Layer*:**

$$\text{Error} = O_{Desired} - O_{Estimated}$$

where,

$O_{Desired}$ is the desired output value of the Node in the Output Layer

$O_{Estimated}$ is the estimated output value of the Node in the Output Layer

Multi-Layer Perceptron Learning Algorithm

Step 2: Backward Propagation

1. **Calculate Error at each node:**

*For each Unit k in the **Output Layer***

$$\text{Error}_k = O_k (1 - O_k) (O_{Desired} - O_k)$$

where,

O_k is the output value at Node k in the Output Layer.

$O_{Desired}$ is the desired output value of the Node in the Output Layer.

*For each unit j in the **Hidden Layer***

$$\text{Error}_j = O_j (1 - O_j) \sum_k \text{Error}_k \, w_{jk}$$

where,

O_j is the output value at Node j in the Hidden Layer.

Error_k is the error at Node k in the Output Layer.

w_{jk} is the weight in the link from Node j to Node k.

Multi-Layer Perceptron Learning Algorithm

2. Update all weights and biases:

Update weights

$$\Delta w_{ij} = \propto \times \text{Error}_j \times O_i$$
$$w_{ij} = w_{ij} + \Delta w_{ij}$$

where,

O_i is the output value at Node i.

Error_j is the error at Node j.

$\propto$ is the learning rate.

w_{ij} is the weight in the link from Node i to Node j.

Δw_{ij} is the difference in weight that has to be added to w_{ij}.

Multi-Layer Perceptron Learning Algorithm

Update Biases

$$\Delta\theta_j = \propto \times \text{Error}_j$$
$$\theta_j = \theta_j + \Delta\theta_j$$

where,

Error_j is the error at Node j.

$\propto$ is the learning rate.

θ_j is the bias value from Bias Node 0 to Node j.

$\Delta\theta_j$ is the difference in bias that has to be added to θ_j.

Backpropagation

- Backpropagation is a method used to train neural networks. It helps the network adjust its weights and biases so that it can make better predictions. Here's how it works:

1. **Forward Pass**: The network makes a prediction based on the current weights and biases.
2. **Calculate Error**: It compares the prediction to the actual answer and calculates how wrong it was (this is called the **loss**).
3. **Backward Pass**: The network figures out how much each weight and bias contributed to the error and adjusts them to reduce the error.

Analogy: Learning to Shoot a Basketball

- Imagine you're learning to shoot a basketball into a hoop. You don't know the right amount of force or angle to use, so you start with a random guess.

1. **Forward Pass**: You throw the ball with your current guess (force and angle).
2. **Calculate Error**: You see where the ball lands compared to the hoop. If it’s too far to the left, you know you need to adjust your aim.
3. **Backward Pass**: You figure out how much to change your force and angle so the next shot is closer to the hoop.

- You repeat this process many times, and each time you get a little better. Backpropagation works the same way for a neural network!

Step-by-Step Explanation with Example

Let's say we have a very simple neural network with:

- **Input**: $x = 2$
- **Weight**: $w = 3$ (initially random)
- **Bias**: $b = 1$ (initially random)
- **True Output**: $y_{\text{true}} = 10$

The network makes a prediction using the formula:

$$y_{\text{pred}} = w \cdot x + b$$

Step 1: Forward Pass

- Compute the prediction:

$$y_{\text{pred}} = 3 \cdot 2 + 1 = 7$$

- The true output is $y_{\text{true}} = 10$, so the error (loss) is:

$$\text{Loss} = (y_{\text{true}} - y_{\text{pred}})^2 = (10 - 7)^2 = 9$$

Step 2: Calculate Gradients

The goal is to reduce the loss by adjusting w and b. To do this, we calculate how much changing w and b affects the loss.

- **Gradient for w**:

$$\frac{\partial \text{Loss}}{\partial w} = 2 \cdot (y_{\text{true}} - y_{\text{pred}}) \cdot x = 2 \cdot (10 - 7) \cdot 2 = 12$$

- **Gradient for b**:

$$\frac{\partial \text{Loss}}{\partial b} = 2 \cdot (y_{\text{true}} - y_{\text{pred}}) = 2 \cdot (10 - 7) = 6$$

Step 3: Update Weights and Bias

We use the gradients to update w and b. Let's assume the learning rate $\eta = 0.1$.

- Update w:

$$w_{\text{new}} = w - \eta \cdot \frac{\partial \text{Loss}}{\partial w} = 3 - 0.1 \cdot 12 = 1.8$$

- Update b:

$$b_{\text{new}} = b - \eta \cdot \frac{\partial \text{Loss}}{\partial b} = 1 - 0.1 \cdot 6 = 0.4$$

Step 4: Repeat

Now, with the updated $w = 1.8$ and $b = 0.4$, we repeat the process:

- New prediction:

$$y_{\text{pred}} = 1.8 \cdot 2 + 0.4 = 4$$

- New loss:

$$\text{Loss} = (10 - 4)^2 = 36$$

Wait, the loss increased! This means the learning rate might be too high. We can reduce the learning rate (e.g., $\eta = 0.01$) and try again.

Why is Backpropagation Important?

- It allows the network to learn from data by minimizing errors.
- It’s the foundation of training deep neural networks.

Advantages

1. **Efficient Learning**:
    1. Backpropagation is computationally efficient compared to other optimization methods, especially for large neural networks.
2. **Handles Complex Models**:
    1. It can train multi-layer neural networks (e.g., MLPs) to model complex, non-linear relationships in data.
3. **General Purpose**:
    1. Works for a wide range of tasks, including classification, regression, and function approximation.
4. **Automatic Feature Learning**:
    1. Unlike traditional machine learning algorithms, backpropagation allows neural networks to automatically learn useful features from raw data.
5. **Scalable**:
    1. Can be applied to very large datasets and networks, especially with modern hardware (e.g., GPUs).
6. **Iterative Improvement**:
    1. Continuously improves the model's performance by minimizing the error over multiple iterations.

Disadvantages

- **Sensitive to Initial Weights**:
  - The performance of backpropagation depends heavily on the initial values of weights and biases. Poor initialization can lead to slow convergence or getting stuck in local minima.
- **Risk of Overfitting**:
  - Neural networks trained with backpropagation can overfit the training data, especially if the network is too large or the dataset is too small.
- **Computationally Expensive**:
  - For very deep networks or large datasets, backpropagation can be slow and require significant computational resources.
- **Vanishing/Exploding Gradients**:
  - In deep networks, gradients can become very small (vanishing) or very large (exploding), making training difficult. This is especially common with activation functions like sigmoid or tanh.

Disadvantages

- **Requires Tuning**:
  - Hyperparameters like learning rate, number of layers, and number of neurons must be carefully tuned, which can be time-consuming.
- **Local Minima**:
  - Backpropagation uses gradient descent, which can get stuck in local minima instead of finding the global minimum of the loss function.
- **Black Box Nature**:
  - Neural networks trained with backpropagation are often considered "black boxes" because it’s hard to interpret how they make decisions.

When to Use Backpropagation

- Use backpropagation when:
  - You have a large dataset.
  - The problem is complex and non-linear.
  - You need high accuracy and can afford computational resources.
- Avoid backpropagation when:
  - The dataset is very small (high risk of overfitting).
  - You need interpretable models.
  - Computational resources are limited.