6CS012 – Artificial Intelligence and Machine Learning.
Lecture – 03
From Machine to Deep Learning
An Introduction to Artificial Neural Network.

Siman Giri {Module Leader – 6CS012}

Learning Objective,

- Discuss on the limitations of Logistic Regression for image classification Task?
  - A brief history of neural networks.
  - Early Models and their limitations.
  - Introduction to Modern Neural Networks.

1. Machine → Deep Learning?

Why?

1.1 Last week : Logistic Regression for Classification.

- We implemented Logistic Regression using ERM Framework:

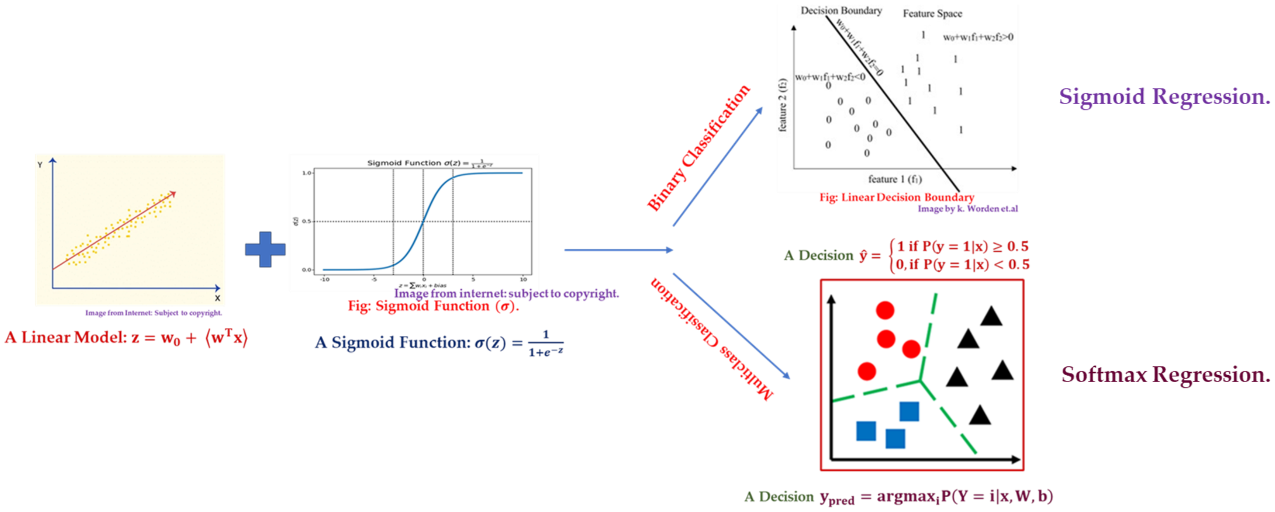


Fig: Logistic Regression for Binary Classification and Multiclass Classification Problem.

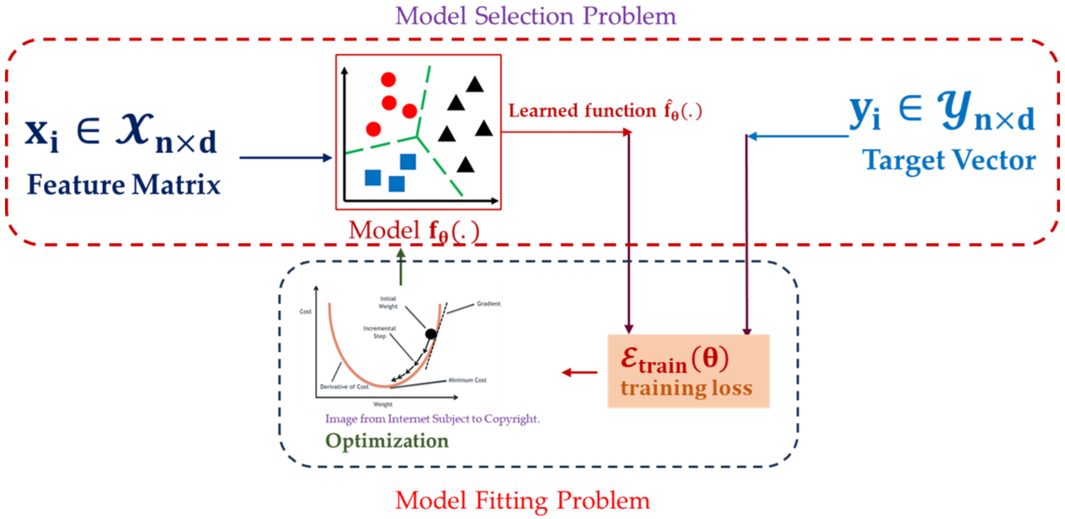


Fig: Framing Learning Problem within ERM framework.

Logistic Regression $\rightarrow$ ERM Framework.

1.1.1 Last week : Dissecting Logistic Regression.

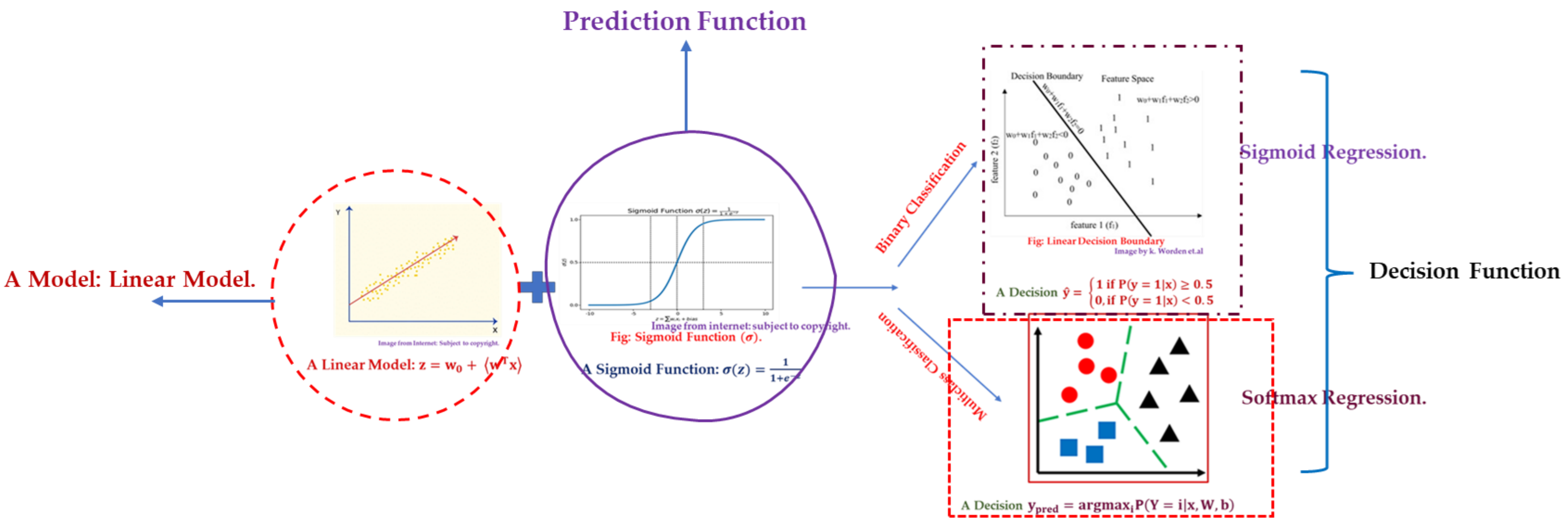
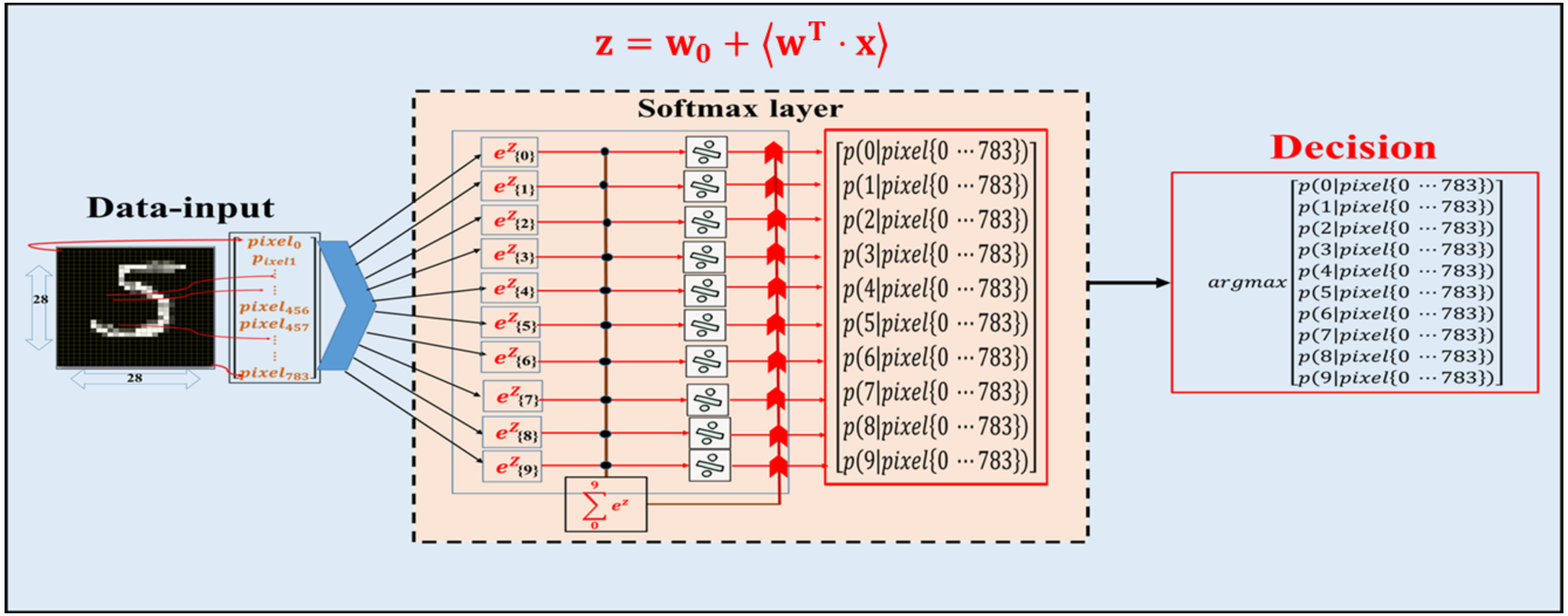


Fig: Logistic Regression for Binary Classification and Multiclass Classification Problem.

Fig: Component of a Logistic Regression Model.

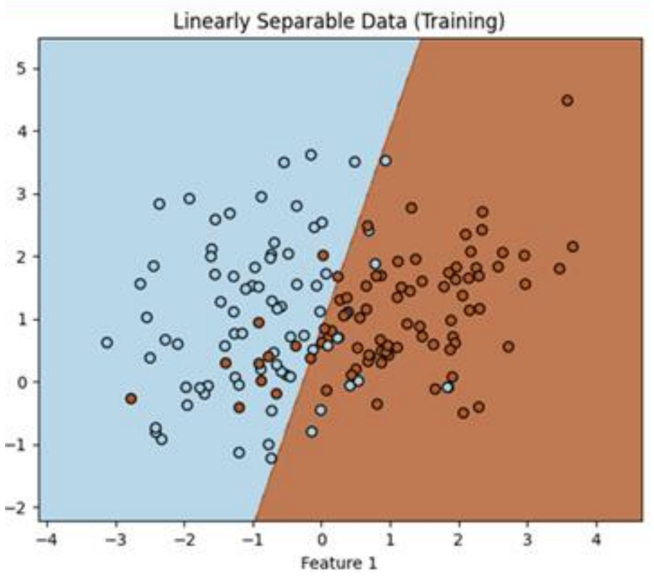
1.1.2 Last week : Application.

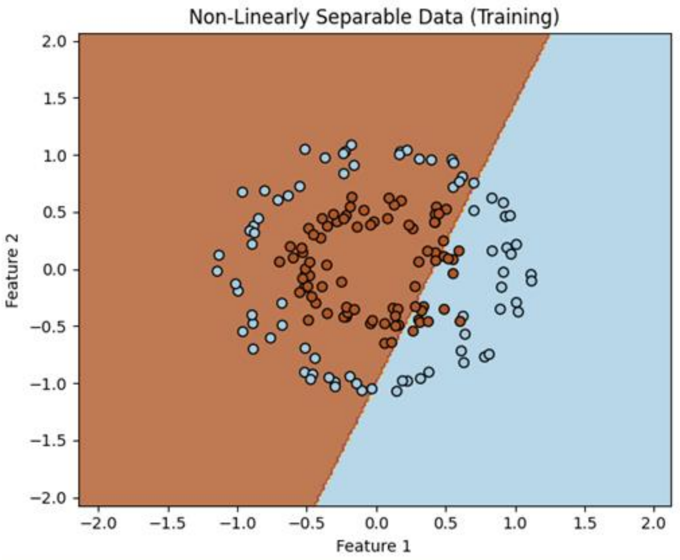
- We use Softmax Regression and Build:
  - "**MNIST Handwritten Digit Classification**."



1.2 Our Observation – 1:

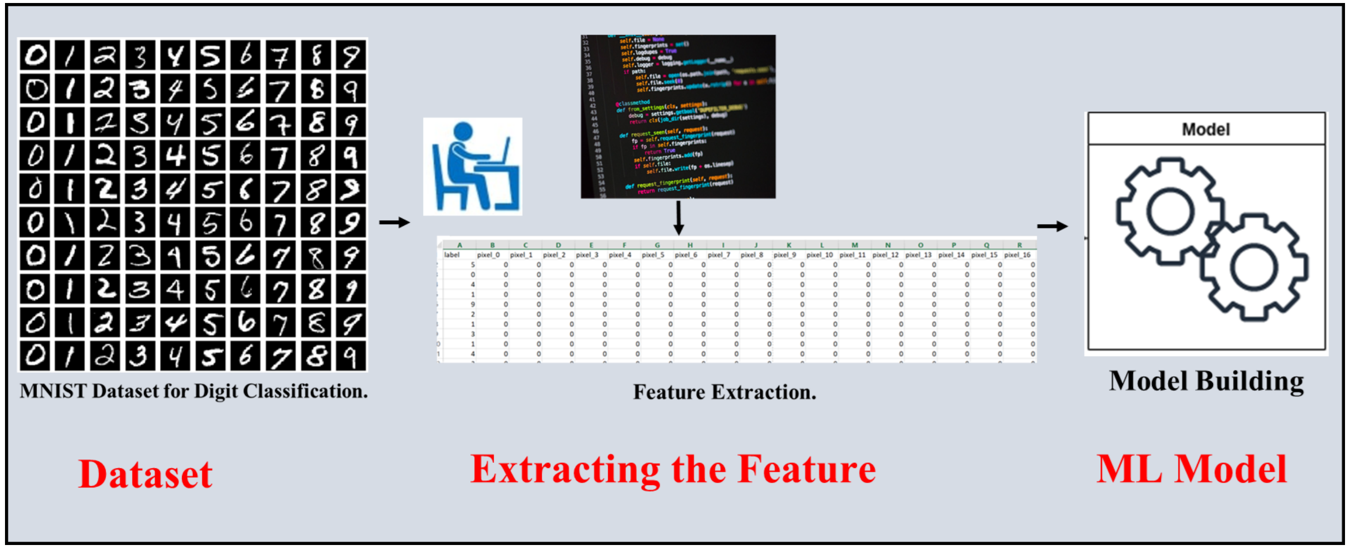
- **Non-Linear Decision Boundary:**
  - **Logistic Regression{Softmax Regression}** was better on separating a data in classes/label those were **linearly separable**.
  - **What for data where classes/label are not linearly separable?**
    - {In **ML there exists more advance algorithm for example DT, SVM etc.** but not suitable for unstructured dataset like **images.**}





1.3 Our Observation – 2:

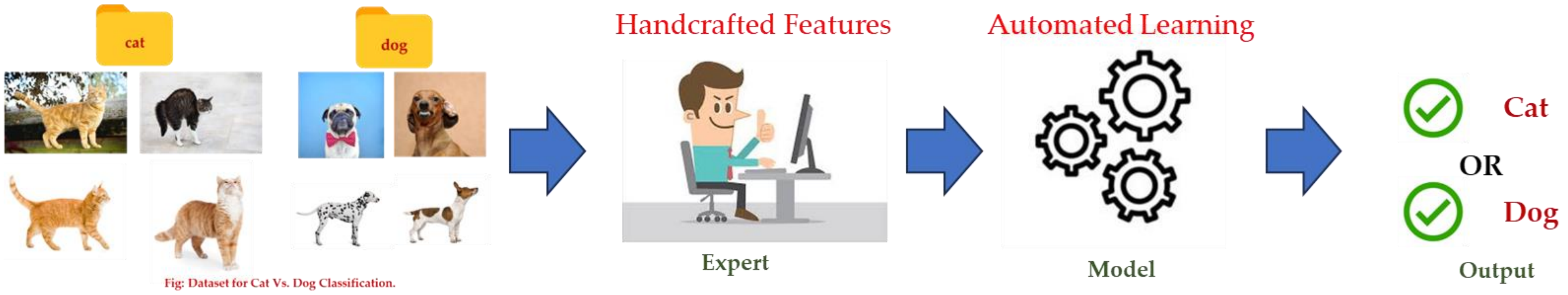
- **Extraction of Features:**
  - When we extracted only pixel values, we got the csv file with 784 columns
    - i.e. very high dimensions , and our dataset was only of the size of 28×28.
  - Imagine for larger images, how big our dataset's dimension may be.
  - We can use feature extraction to reduce some dimension. What is the challenge?



Why Feature extraction is a Challenge?

1.4 Machine Learning Vs. Deep Learning.

- **Machine Learning:**
  - One crucial aspect of machine learning approaches to solving problems is that **human** and often undervalued **engineering** plays an **important role**.
    - A human still has to frame the problem,
      - **acquire and organize data, design a space of possible solutions**,
      - **select a learning algorithm and its parameters**,
      - **apply the algorithm to the data**,
      - **validate the resulting solution to decide whether it's good enough to use, etc**.
  - These steps are of great importance.

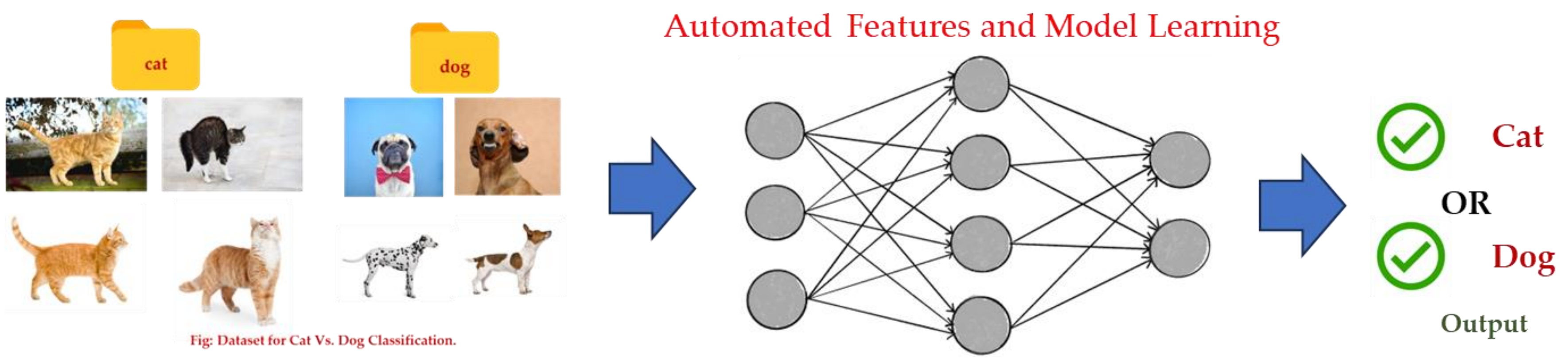


1.4.1 Why Feature Extraction is a Limitations?

- **Manual Effort & Expertise Required:**
  - Traditional methods rely on handcrafted features such as edge detectors , texture descriptors , or color histograms.
  - Engineers and researchers must experiment with different feature extraction techniques to find the best ones for specific tasks.
- **Suboptimal Performance:**
  - Handcrafted features may not always capture the most relevant information in complex images.
  - They are often designed for specific datasets and may not generalize well across different conditions (e.g., lighting variations, occlusions).
- **Curse of Dimensionality:**
  - Extracted features can be high-dimensional, making it computationally expensive to process and requiring dimensionality reduction techniques like PCA.
  - Irrelevant or redundant features can degrade model performance.
- **Lack of Adaptability:**
  - Features designed for one problem (e.g., object classification) might not be suitable for another (e.g., object detection or segmentation).
  - Traditional feature extraction does not automatically adapt to variations in image quality, background, or perspective.

1.4 Machine Learning Vs. Deep Learning.

- Can we learn the underlying features directly from data?
  - Yes, Deep Learning.
    - { We will talk more on this as we move forward with more complex deep learning algorithms.}



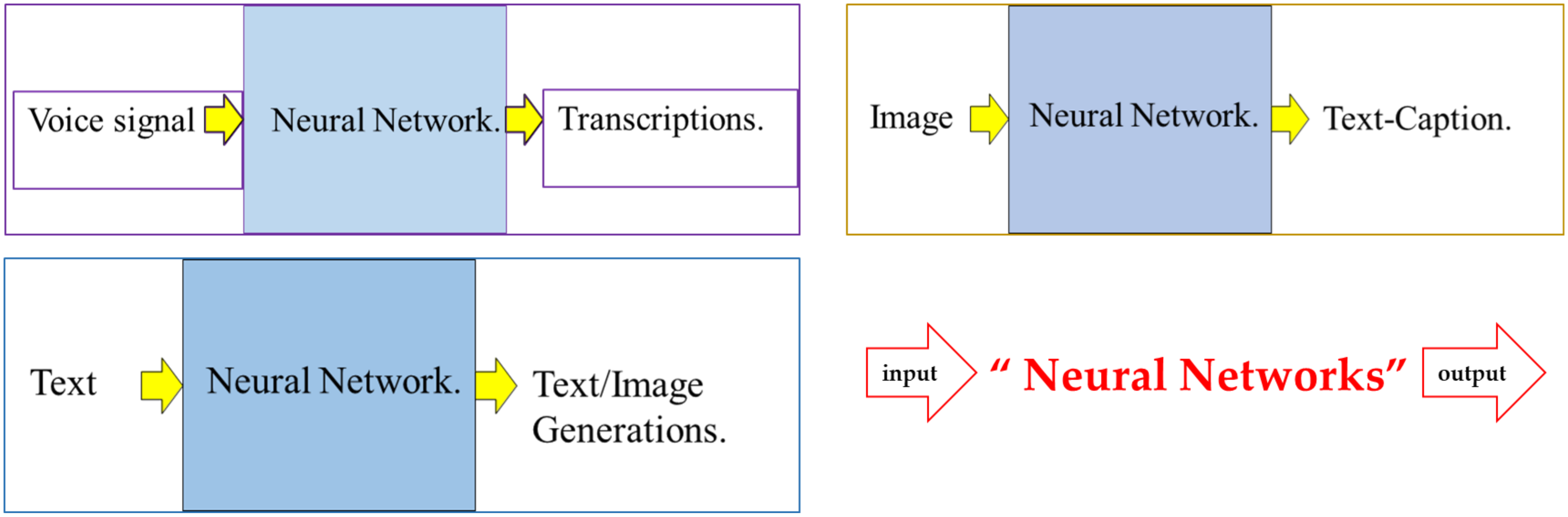
1.5 Modern Breakthrough of Deep Learning.

- Deep learning have become one of the main approaches to AI
    - They have been successfully applied to various pattern recognition, prediction, and analysis problems
- In many problems they have established the state of the art
    - Often exceeding previous benchmarks by large margins
        - Sometimes solving problems, you couldn't solve using earlier ML methods

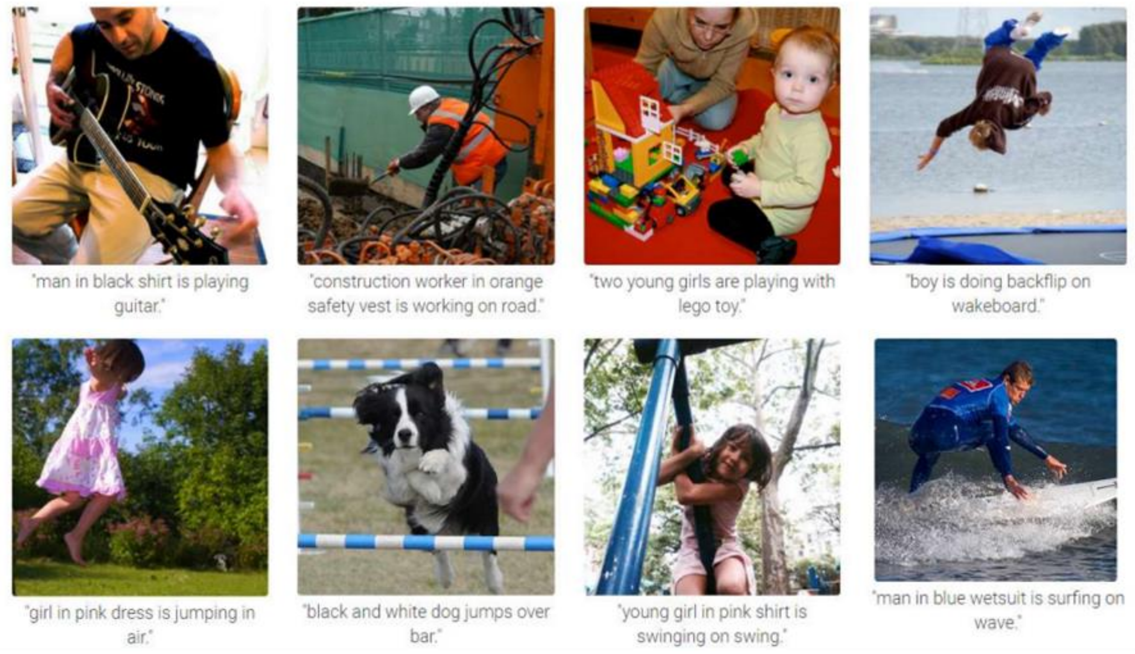
Some Modern Breakthroughs Of Deep Learning.

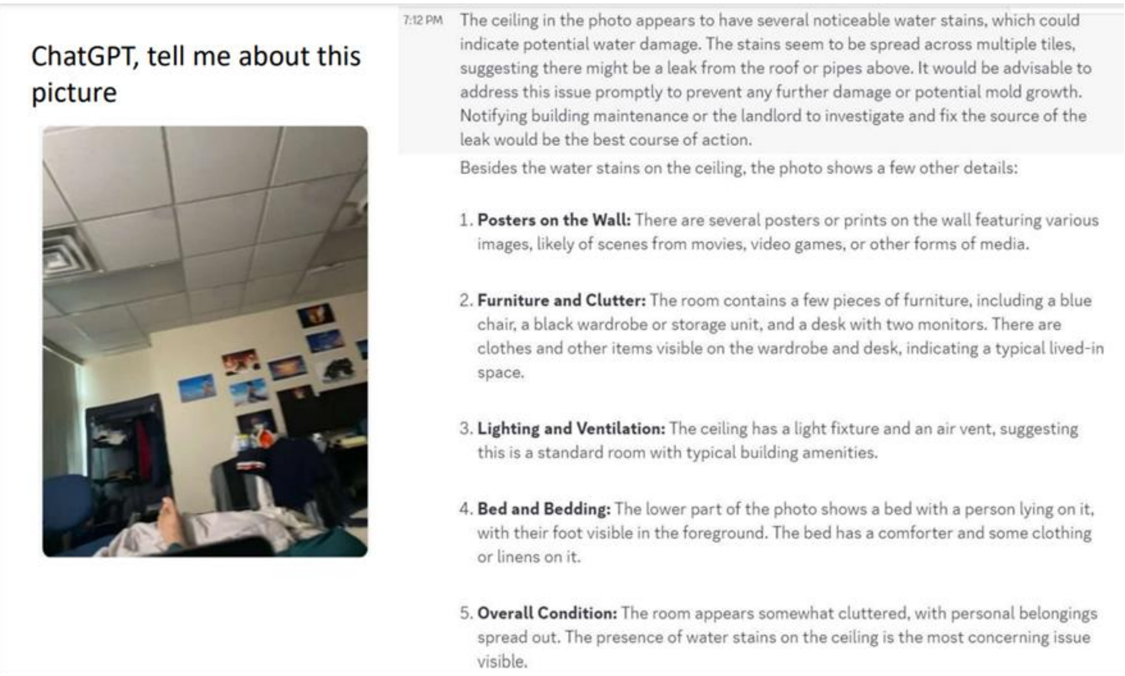
What is Common among all above

1.6 The Common Elements in above Breakthrough.



1.7 Neural Networks are Taking Over.

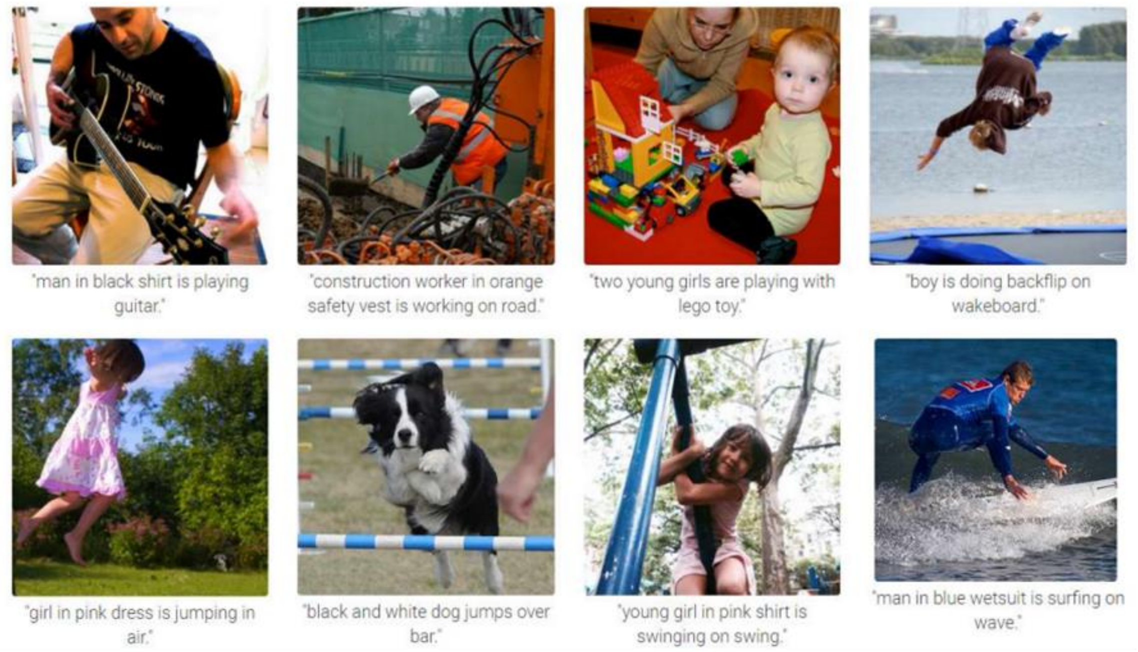




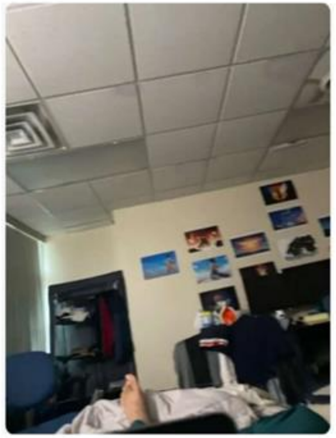
Auto Caption Generator → **Auto Description Generator**

What are Neural Networks?

1.7 Neural Networks are Taking Over.



ChatGPT, tell me about this picture



7:12 PM The ceiling in the photo appears to have several noticeable water stains, which could indicate potential water damage. The stains seem to be spread across multiple tiles, suggesting there might be a leak from the roof or pipes above. It would be advisable to address this issue promptly to prevent any further damage or potential mold growth. Notifying building maintenance or the landlord to investigate and fix the source of the leak would be the best course of action.

Besides the water stains on the ceiling, the photo shows a few other details:

1. **Posters on the Wall:** There are several posters or prints on the wall featuring various images, likely of scenes from movies, video games, or other forms of media.
2. **Furniture and Clutter:** The room contains a few pieces of furniture, including a blue chair, a black wardrobe or storage unit, and a desk with two monitors. There are clothes and other items visible on the wardrobe and desk, indicating a typical lived-in space.
3. **Lighting and Ventilation:** The ceiling has a light fixture and an air vent, suggesting this is a standard room with typical building amenities.
4. **Bed and Bedding:** The lower part of the photo shows a bed with a person lying on it, with their foot visible in the foreground. The bed has a comforter and some clothing or linens on it.
5. **Overall Condition:** The room appears somewhat cluttered, with personal belongings spread out. The presence of water stains on the ceiling is the most concerning issue visible.

Auto Caption Generator → **Auto Description Generator**

What are Neural Networks?

It is said to be inspired by "*human ability of thinking*".

2. Introduction to Neural Networks.

{From Biological Inspiration to Computational Models.}

2.1 Way before Neural network …

- Humans have the remarkable ability to:
  - Learn and Solve problems
  - Recognize patterns and Memorize
  - Create
  - Think deeply
- but how exactly do humans function or think?
  - Can we emulate i.e. put into machine?

2.2 Early Models of Human Cognition.

- **Associationism (– 400 BC – 1900 AD by Aristotle, Plato, David Hume, Ivan Pavlov, …)**
  - 360 BC: Aristotle wrote:
    - "*Hence, too, it is that we hunt through the mental train, excogitating from the present or some other, and from similar or contrary or coadjacent. Through this process reminiscence takes place. For the movements are, in these cases, sometimes at the same time, sometimes parts of the same whole, so that the subsequent movement is already more than half accomplished*."
      - **In English: We memorize and rationalize through association.**
- Theory of Associationism:
  - "*Learning is a mental process that forms associations between temporally related phenomena*."
    - Example: "hey here's a bolt of lightning, we're going to hear thunder"
- Challenge: **But where are the associations stored and how?**

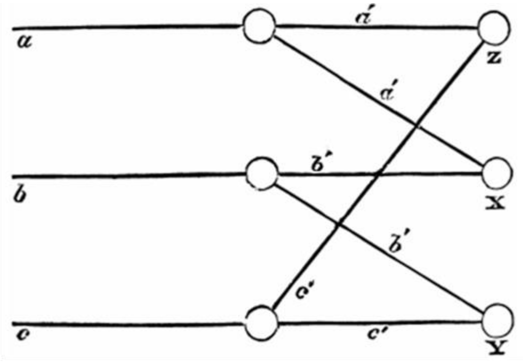
2.3 Beginning the Era of Connectionism.

- {Cautions: we have lots of historical development that happened and will discuss major events that happened in Mid 1800s.}
- **Alexander Bain**, in 1873 in his work "Mind and Body" proposed that
  - "The information is in the connections" suggesting:
    - The brain is a mass of interconnected neurons.
    - He floated the idea: The brain is a network of neurons, and
      - collection of neurons **excites and simulate each other** and **the memory** are stored in the **connections**.
        - different combinations of **inputs** can result in different **outputs**.

Alexander Bain

The Brain

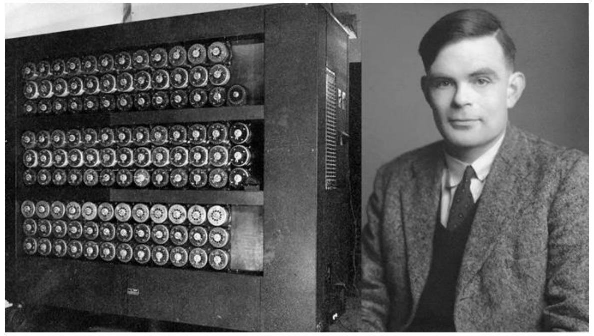
Brain: Interconnected Neurons.



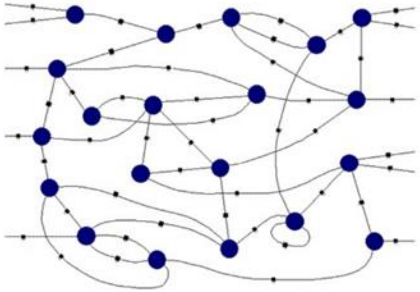
Bain's Idea of Neural Grouping.

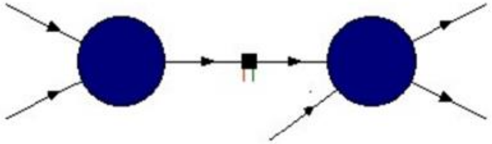
2.3.1 Connectionist Machines

- Neural networks **are connectionist machines** as opposed to **Von – Neumann Machines**.
- The machine has many **non-linear processing units**
  - The program is the connections between these units
    - Connections may also define memory



Alan Turing with his Turing Machine.



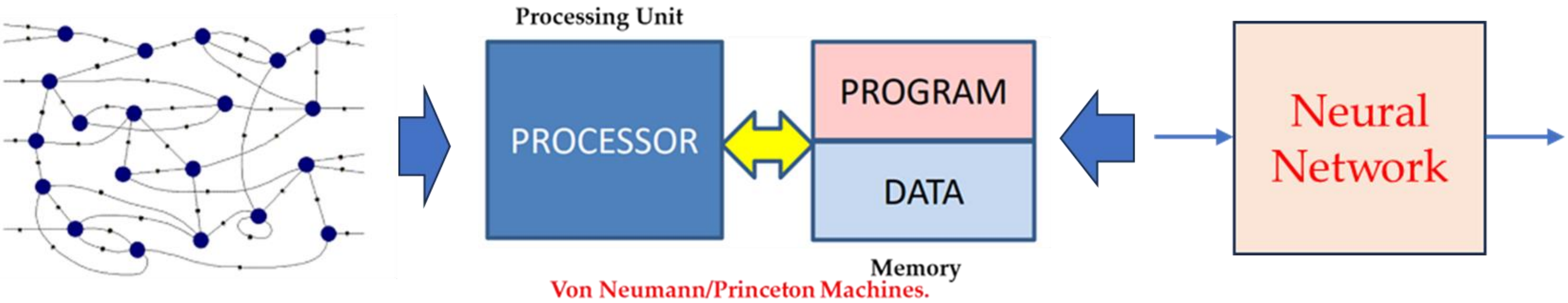


Alan Turing's Connectionist Model - 1948

But what are these independent processing units?

2.3.2 Connectionist Machines: Warning

- Current deep learning algorithms emulate neural networks but run on
  - **Von Neumann machines**, which have a fundamentally different architecture from biological brains.
  - This creates both **strengths and limitations** in modern AI systems.



We emulate neural network as connections in Von Neumann Machines.

2.4 Modelling the Brain: Neurons.

- What are the units?
  - Answer: A neuron.
- The structure of Biological Neurons {major elements}:
  - **Dendrite**: receives signals from other neurons
  - **Synapse:** point of connection to other neurons
  - **Soma:** process the information
  - **Axon:** transmits the output of this neuron
- How neurons transmits the information?
  - *There are almost billions neurons*.

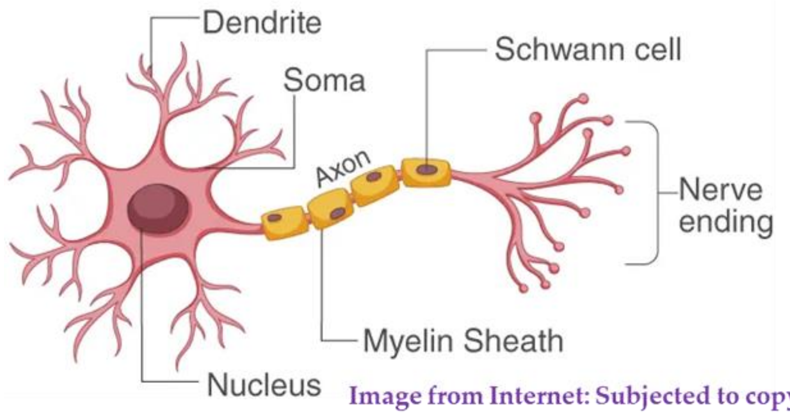


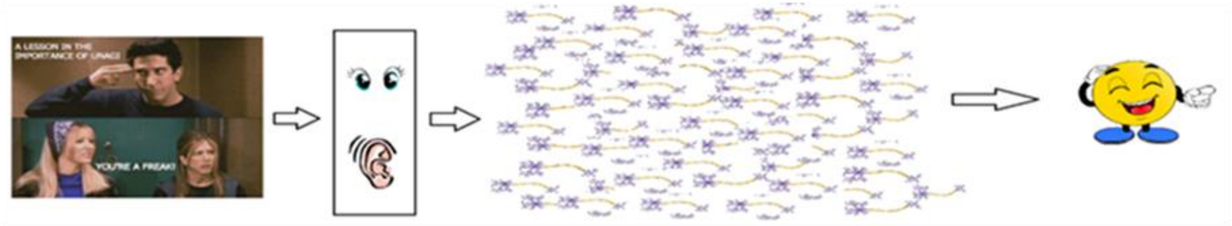
Image from Internet: Subjected to copyright.

Fig: Structure of a Neuron.

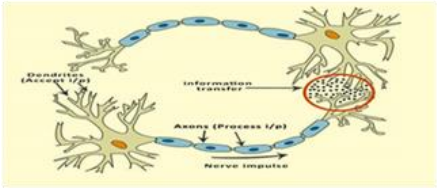
2.5 Modelling the Brain: Connectionist.

- How neurons transmits the information?
  - A very cartoonish illustrations:

  - There are 100 Billion Neurons;



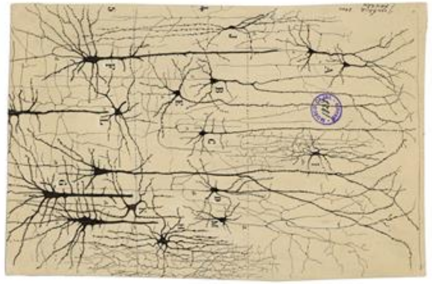
  - Do they all act at once:



What are there Mathematical or Computational Representations?
How can we model them?

2.6 Biological Neurons to Computational Model.

- **Observations from working of Neurons:**
  - Interconnected network of hundreds to thousands of neurons works in parallel
  - For neurons to **activate** or fire or spike **certain threshold** must be passed
  - **Inputs** are received and can be of:
    - **Excitatory inputs/Synapses**:
      - **Transmits** input to the neuron.
    - **Inhibitory inputs/Synapses:**
      - Any signals from an inhibitory inputs **prevents** neuron from firing, regardless of other inputs.



Drawing: Ramon Cajal.

2.6.1 Biological Neurons to Computational Model.

- **Neurons as Biological Computational devices:**
  - How can we **model the single neuron** based on the above observation?
    - A Neuron can either fire or not-fire,
      - thus, output of neuron can be **binary i.e. 0 or 1**
    - As output can only be 0 or 1,
      - if we can make **collection of input also be binary**,
        - we can create a **threshold function using a logic operator**.
- Our First computational model was based on above thought and
  - was called **McCulloch & Pitts Neuron**.

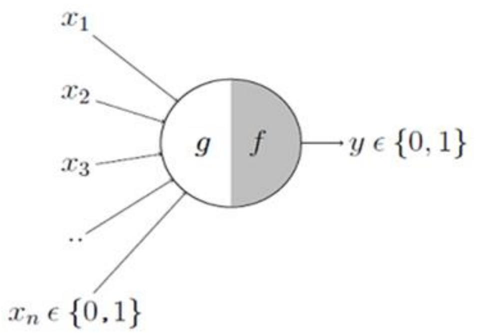
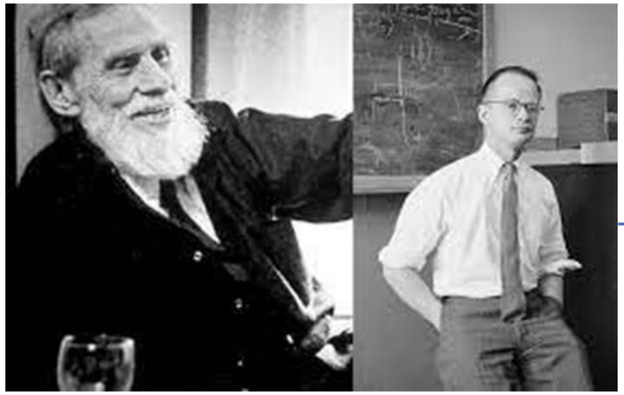


Fig: A Symbolic representation of "How we want our Neuron to be?"

Warren Sturgis McCulloch{1898 – 1969}: Neurophysiologist, logician and Hobo.



Walter Pitts {1923 – 1969}: Neurophysiologist, and Mathematician.

3. McCulloh and Pitts Neuron.

{Earlier Computational or Mathematical Representations of Neuron.}

3.1 The McCulloch – Pitts Artificial Neuron.

- The first computational model of a neuron was proposed by Warren **MuCulloch** (neuroscientist) and **Walter Pitts** (logician) in 1943 in their paper titled:
  - *"Logical Calculus of Ideas Immanent in Nervous Activity* (1943)."
  - Based on there thought and previous work on neural nets:
    - on what was the **fundamentals elements** to represent **computation** in **biological neurons**
    - they proposed a model also known as
      - **linear threshold gate or threshold logic units** or **MCP neurons.**
      - Modeled the neurons of the brain (and the brain itself) as performing propositional logic, where each neuron evaluates the truth value of its input (**propositions**)
      - Effectively Boolean logic aka synaptic model

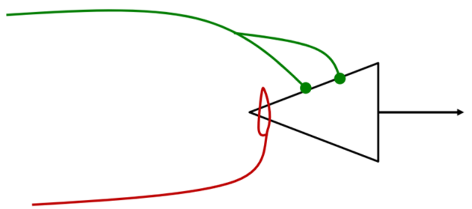


Fig: A Single MCP Neuron proposed as Synaptic Model.

3.2 An MCP Neuron.

- **Excitatory synapse:**
  - Transmits weighted input to the neuron
- **Inhibitory synapse:**
  - Any signal from an inhibitory synapse prevents neuron from firing
  - The activity of any inhibitory synapse absolutely prevents excitation of the neuron at that time.
    - Regardless of other inputs

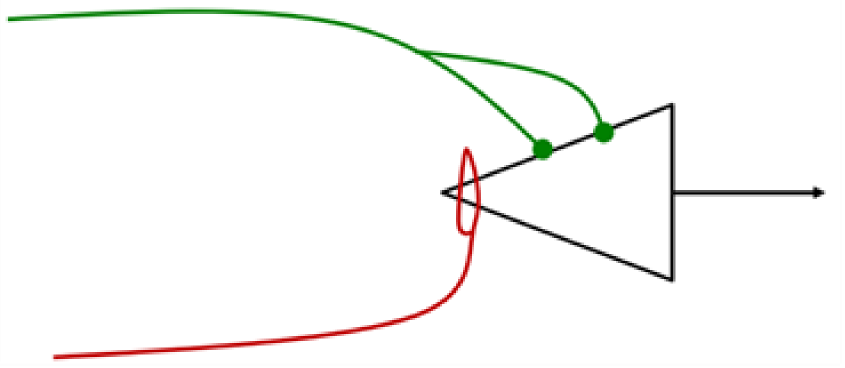
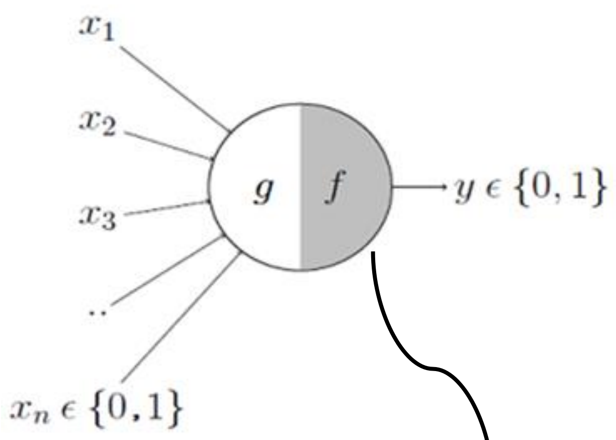


Fig: A Single MCP Neuron proposed as Synaptic Model.

3.2.1 Computational Model of MCP Neuron.

- **Mathematical Formalization of MCP.**
  - MCP describes the activity of single neuron with **two states:** firing(1) or not firing(0).

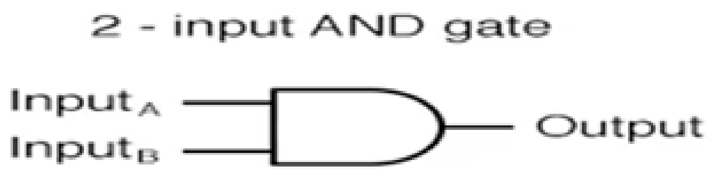
For input $x_i \in \{0, or\ 1\}$ and output $y_i = \{0\ or\ 1\}$
{
    $y = 0$ if any x_i is inhibitory,
else:
    {
    $g(x_1, x_2, x_3 \ldots, x_n) = g(X) = \sum_{i=1}^{n} x_i$
    {
    if: $g(X) \geq T$
        $y = f(g(X)) = 1$
    else: $g(X) < T$
        $y = f(g(X)) = 0$
    }



We will use this notation for easy representations.

3.3 MCP Neuron and Boolean Operations.

- For emulating **AND** Functions:
  - The ***AND* function** is "**activated**" only when all the **incoming inputs** are "**on**",
    - this is, ***it outputs a 1 only when all inputs are 1***.
  - What will be my **Threshold Function**?



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

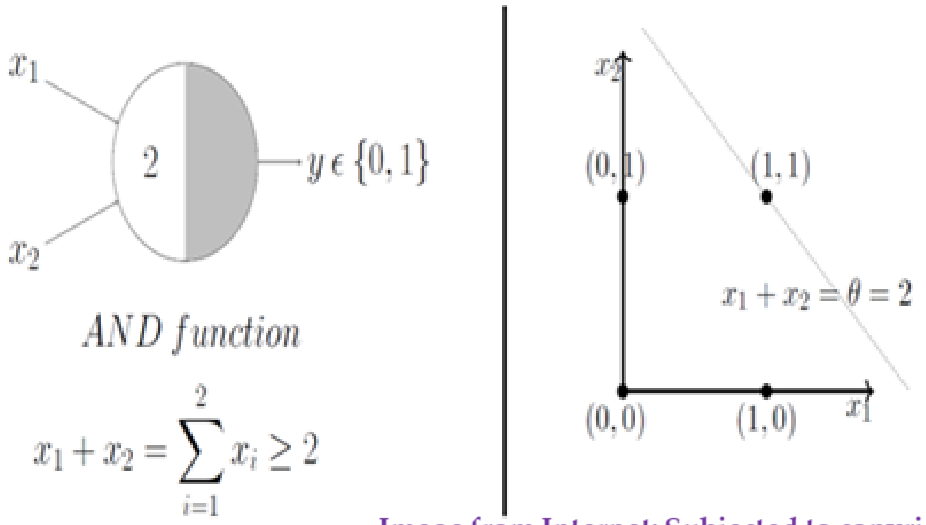
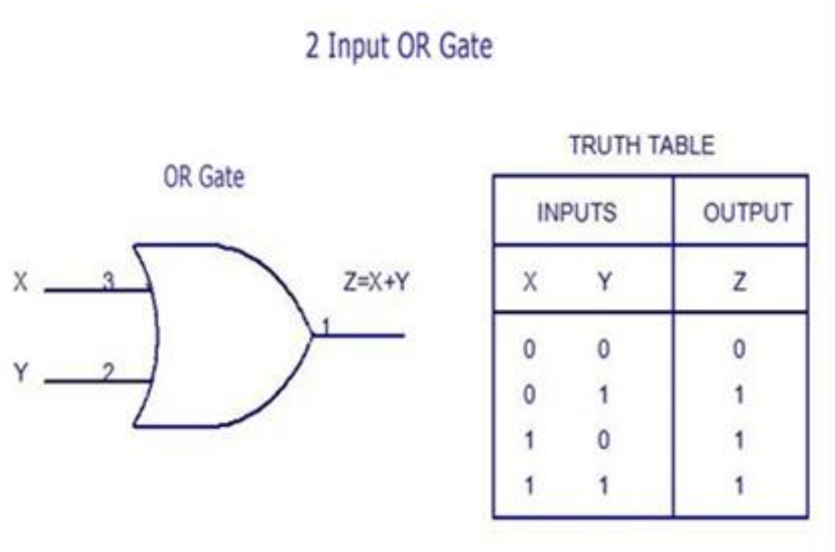


Image from Internet: Subjected to copyright.

3.3 MCP Neuron and Boolean Operations.

- For OR Function:
    - The ***OR function*** is "**activated**" when ***at least one*** of the **incoming inputs** is "**on**".
    - What will be my **Threshold Function**?



INPUTS		OUTPUT
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

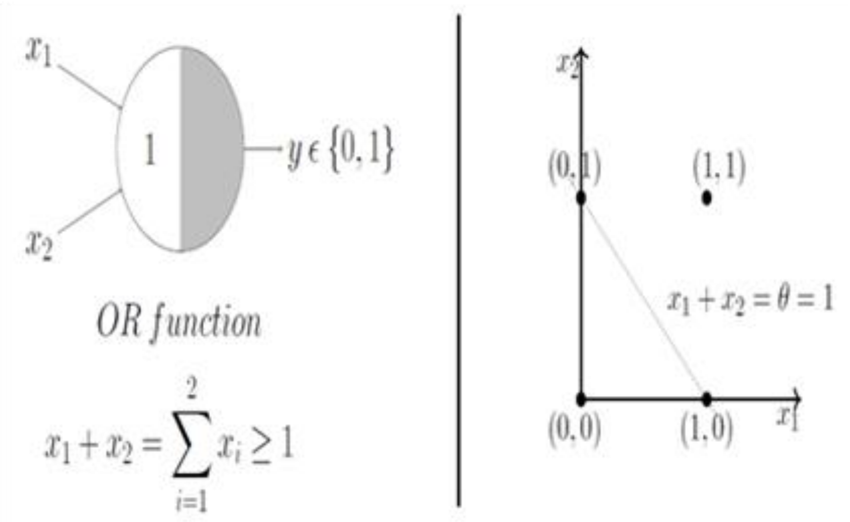


Image from Internet: Subjected to copyright.

3.3 MCP Neuron and Boolean Operations.

- For **XOR Function**:
  - The ***XOR (Exclusive OR) function*** is **activated** (outputs "on" or 1)
    - when **exactly one** of the **incoming inputs is** "**on**" (1), **but not both**.
  - What will be my **Threshold Function**?

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

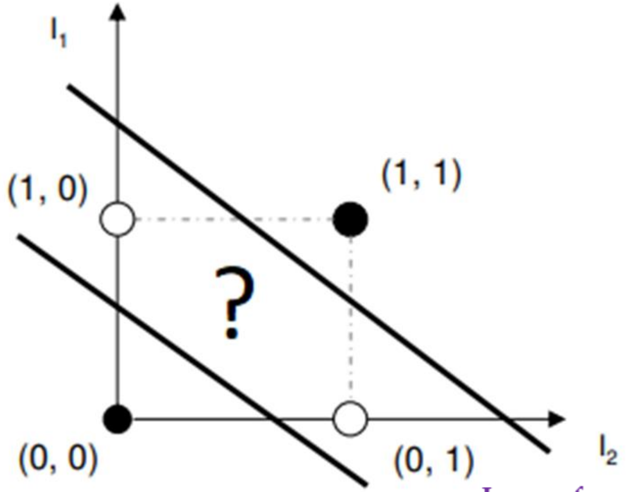


Image from Internet: Subjected to copyright.

- The phenomenon also called "**XOR Problem**" or "**XOR Realizations**".

3.4 Limitation of MCP Neurons.

- A single **McCulloch Pitts Neuron** can be used to represent boolean functions which are linearly separable.
  - **Linear separability** (for boolean functions) :
    - There exists a line (plane) such that
      - all inputs which produce a 1 lie on one side of the line (plane)
      - and all inputs which produce a 0 lie on other side of the line (plane)
- The **MCP Neuron Architecture** lacks **several characteristics** of **biological networks**:
  - **Complex connectivity patterns i.e. represents single neuron only**
  - **Processing of continuous values**
  - **A measure of importance**
  - **A learning procedure**
- Regardless of limitation, MCP is considered **a significant** first step towards development theory of "Mind and Body" laying the foundations for
  - Study of various learning mechanisms and paved the way for advances in artificial intelligence, including the invention of the perceptron and more complex model.

4. Towards a Better Model.

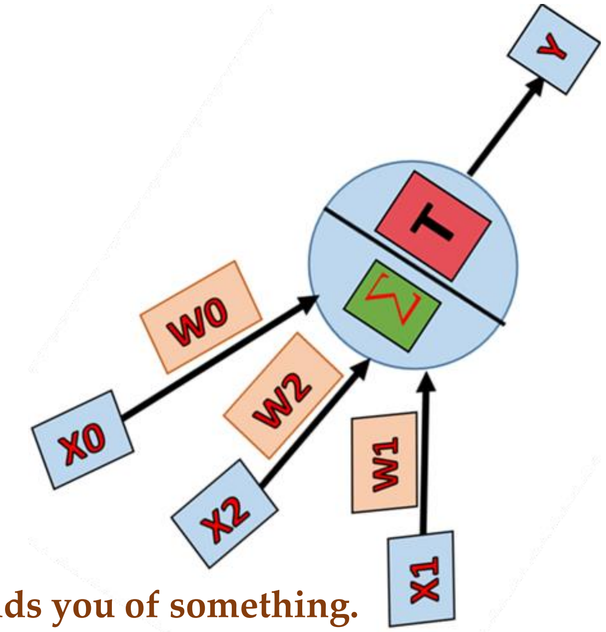
{Introduction to Perceptron and Perceptron Learning Theory.}

4.1 The Perceptron.

- **Frank Rosenblatt**
  - **Psychologist, Logician**
  - ***“ Inventor of the solution to everything, aka the Perceptron (1958) ”.***
- **“the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence,” New York Times (8 July) 1958.**
- **“Frankenstein Monster Designed by Navy That Thinks,” Tulsa, Oklahoma Times 1958.**

4.2 Idea Behind The Perceptron Model.

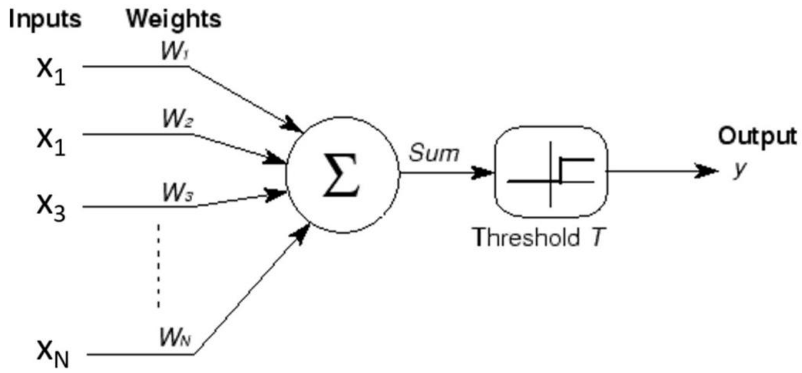
- The **Perceptron**, introduced by **Frank Rosenblatt** and later corrected by **Minsky and Papert**, extends the **McCulloch-Pitts neuron**
  - by incorporating **learnable numerical weights** and
  - an **adaptive learning process**.
  - It serves as a **Linear Binary Classifier**, dividing data into two categories based on a linear decision boundary.
- A perceptron makes a decision based on a **weighted sum** of **inputs**:
  - $\sum_{i=1}^{n} w_i x_i \geq T$
  - here:
    - $w_i \rightarrow$ **learnable weights.**
    - $x_i \rightarrow$ **inputs.**
    - $T \rightarrow$ **learnable Thresholds.**
  - Let's say **I do not want to prefixed my threshold**,
    - instead **learn along with weights**, then we can re-write the equation as:
      - $\hat{y} = \begin{cases} 1 \text{ if } w_o + w_i x_i \geq 0 \\ 0 \text{ if } w_o + w_i x_i < 0 \end{cases}$ $[\text{replace} \rightarrow T = -w_o]$



Reminds you of something.

4.3 Simplified Computational Representation of Perceptron.

- **Mathematical Formulation:**
  - A perceptron takes a set of inputs $x_1, x_2, \dots, x_n$, assigns them weights $w_1, w_2, \dots, w_n$ and computes a weighted sum:
    - $\mathbf{z = \sum_{i=1}^{n} w_i x_i + w_o}$
    - here:
      - $\mathrm{w_i}$ → weights learned during training.
      - $\mathrm{w_o}$ → bias also learned during training. (adjusts the decision boundary)
      - z → net weighted input.
- The perceptron model then applies a threshold activation function (aka step function) on **z (net weighted input)** given by:
  - $\mathbf{f(z) = \begin{cases} 1 \text{ if } z \geq 0 \\ 0 \text{ if } z < 0 \end{cases}}$
  - This **threshold function** determines whether the perceptron activates (outputs 1) or remains inactive (outputs 0).



4.4 Perceptron Learning Algorithm.

- **Input:**
  - Training Dataset $\mathcal{D} = \{(\mathbf{x_i}, \mathbf{y_i})\}$ where:
    - $\mathbf{x_i} =$ **input feature vector**
    - $\mathbf{y_i} =$ **class label** $\in \{\mathbf{0}, \mathbf{1}\}$
  - Learning rate: $\boldsymbol{\eta} = \mathbf{0.01}$ *or* small positive value
  - Number of iterations.
- **Steps:**
  - 1. Initialize weights randomly:
    - Assign small random values to
      - $\mathbf{w} = (\mathbf{w_1}, \mathbf{w_2}, \ldots \mathbf{w_n} \textbf{ and } \mathbf{w_o})$
  - 2. Repeat for each epoch until convergence:
    - For each training sample $(\mathbf{x_i}, \mathbf{y_i})$:
    - Compute the weighted sum:
      - $\mathbf{z} = \sum \mathbf{w_i x_i} + \mathbf{w_o}$
    - Apply the activation function (step function):
      - $\hat{\mathbf{y}} = \begin{cases} \mathbf{1 \text{ if } z \geq 0} \\ \mathbf{0 \text{ if } z < 0} \end{cases}$
- Update the weights if misclassified $(\mathbf{y} \neq \hat{\mathbf{y}})$:
  - $\mathbf{w_i} = \mathbf{w_i} + \boldsymbol{\eta}(\mathbf{y} - \hat{\mathbf{y}})\mathbf{x_i}$
  - $\mathbf{w_o} = \mathbf{w_o} + \boldsymbol{\eta}(\mathbf{y} - \hat{\mathbf{y}})$
- Since both $(\mathbf{y} \textbf{ and } \hat{\mathbf{y}})$ can take only binary values, the term $\mathbf{y} - \hat{\mathbf{y}}$ can be interpreted as:
  - **when** $\mathbf{y} - \hat{\mathbf{y}} = \mathbf{1} \rightarrow$ the weights are increased to push towards correct classification.
  - **when** $\mathbf{y} - \hat{\mathbf{y}} = -\mathbf{1} \rightarrow$ the weights are decreased to correct the misclassification.
  - **when** $\mathbf{y} - \hat{\mathbf{y}} = \mathbf{0} \rightarrow$ no update is needed.
- Stop if no updates occur in an epoch (i.e. convergence).
- Output:
  - Learned weights.

4.4 Perceptron Learning Algorithm.

- Putting it all together:

Perceptron Learning Algorithm with Random Initialization

Algorithm 1 Perceptron Learning Algorithm

Require: Training dataset $D = \{(x_i, y_i)\}$, where:
  x_i is the input feature vector (including bias term)
  $y_i \in \{0, 1\}$ or $y_i \in \{-1, 1\}$
Require: Learning rate η (small positive value, e.g., 0.01)
Require: Number of epochs (max iterations)
  Initialize: Randomly assign small values to $w = (w_1, w_2, \ldots, w_n)$
  Randomly initialize bias b (or include it in w)
  for each epoch **do**
    ConvergenceFlag = **True**
    for each training sample (x_i, y_i) **do**
      Compute weighted sum:

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

      Apply activation function (step function):

$$\hat{y} = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0 \end{cases}$$

      if $\hat{y} \neq y$ **then** ▷ Update weights if misclassified
        Update weights:

$$w_i = w_i + \eta(y - \hat{y})x_i$$

        Update bias:

$$b = b + \eta(y - \hat{y})$$

        ConvergenceFlag = **False**
      end if
    end for
    if ConvergenceFlag is **True then**
      Break ▷ Stop if no updates occur (convergence)
    end if
  end for
  Output: Learned weights w and bias b

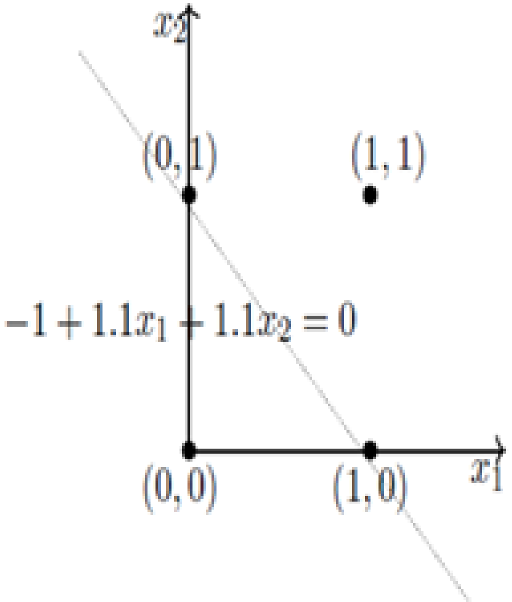
4.5 Perceptron for learning “OR” Function.

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$
$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 > -w_0$$

One possible solution is

$w_0 = -1,\ w_1 = 1.1,\ w_2 = 1.1$

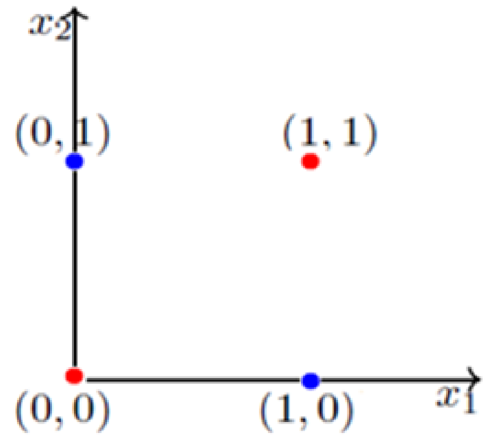


Looks like It works for OR Function.
Let’s try for XOR function.

4.5.1 Perceptron for learning "XOR" Function.

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$
$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 < -w_0$$

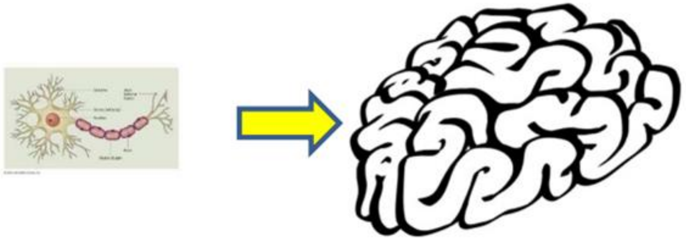


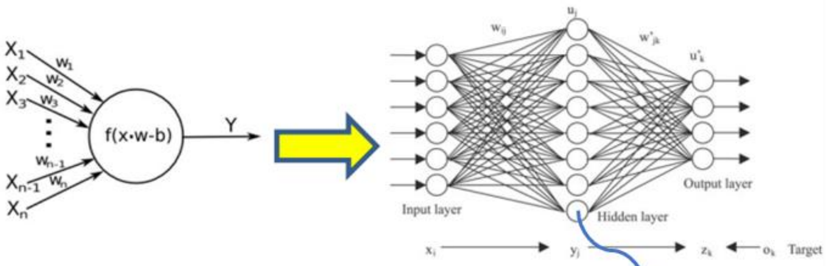
- There does not **exist any possible values** for **w_i and w_o** that can **satisfy all above equation**.
- If we forced to learn using perceptron learning algorithm it may not converge or will terminate with very high error.

This Phenomenon is called Minsky and Papert Correction.

4.5.2 Minsky and Papert Correction.

- Works for linearly separable? XOR is still a problem
- In the book published by Minsky and Papert in 1969, the authors implied that,
  - "*since a single artificial neuron is incapable of implementing some functions such as the XOR logical function, larger networks also have similar limitations, and therefore should be dropped. Later research on three-layered perceptrons showed how to implement such functions, therefore saving the technique from obliteration*."
- The conclusion from Minsky and Papert Correction:
  - Individual elements are weak computational elements.
  - Network of elements are required.





We will discuss more about this next week.

4.6 Solving the XOR Problem.

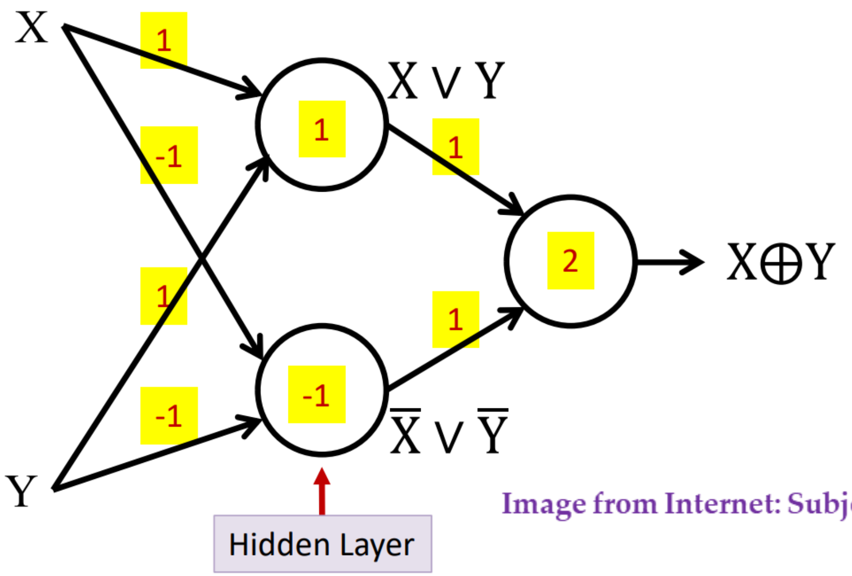
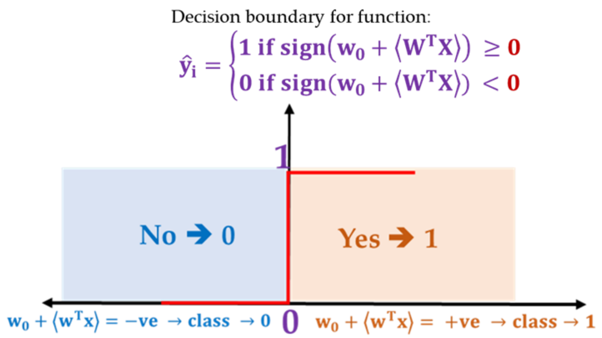


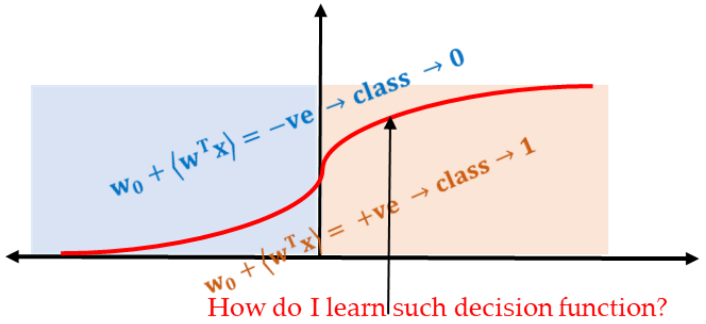
Image from Internet: Subjected to copyright.

- A **multi layer perceptron** could solve for XOR problem:
  - As it can compose arbitrarily complicated Boolean functions!
    - In cognitive terms: Can compute arbitrary Boolean functions over sensory input
    - We will verify this in Tutorial with some exercise so do not miss the Tutorial.
- More on this in the next week

4.7 Perceptron for Real Value Inputs.

- **Step function** is unsuitable for real-valued input due to non-differentiability and binary output, which hinder learning and fine-tuning.
- The **step function** may not capture the subtleties of the data;
  - it creates a sharp boundary at 0,
    - meaning that all positive sums are classified as class 1
    - and all negative sums are classified as class 0.
  - However, there may be cases where net negative values also belong to class 1.





4.8 Sigmoid Neurons.

- **Sigmoid neurons** are a type of artificial neuron that uses the
  - **sigmoid "activation" function** to model the output of the neuron.
  - The sigmoid function is a smooth, S-shaped curve that maps any
    - real-valued input to a value between 0 and 1,
      - making it especially useful for problems that involve probabilities or binary classification.
- The **sigmoid activation function** is defined as:
  - $\sigma(z) = \frac{1}{1+e^{-z}}$
    - here: $z = w_o + w_i x_i$ → the weighted sum of the inputs to the neuron.

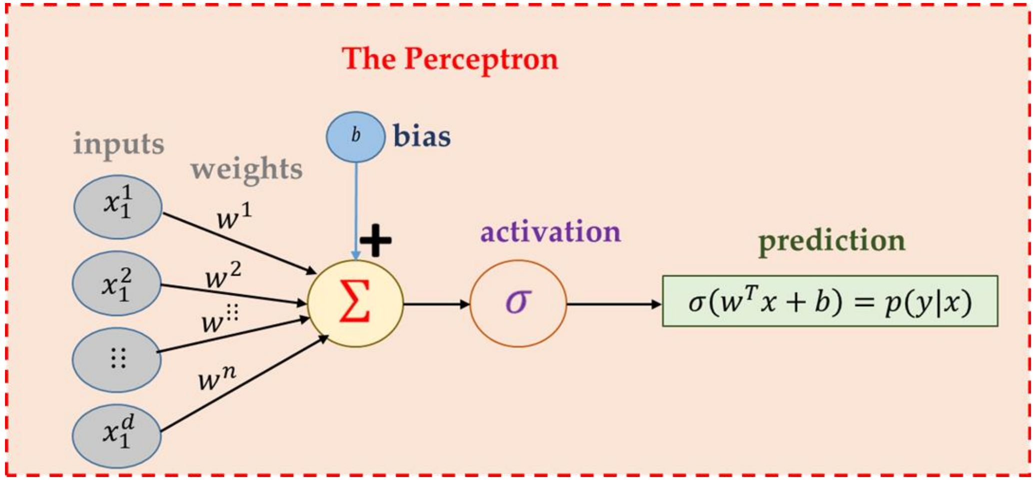
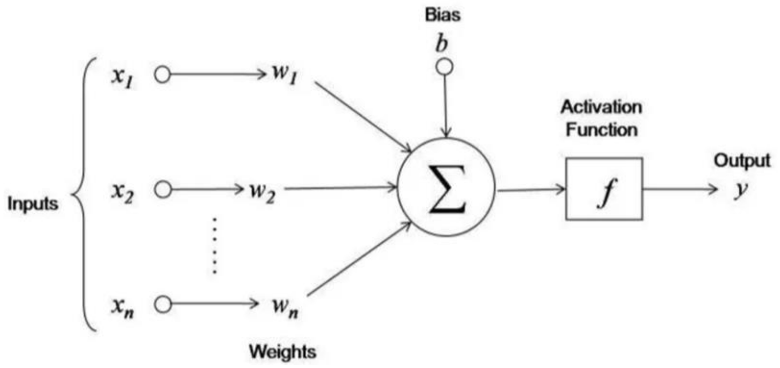


Fig: A single unit of Standard Perceptron with sigmoid.

4.9 The activation function

- In General, Activation Functions are group of function that introduces **non-linearity** to the **output** of neurons.
  - The activation function does the **non-linear transformation to the input**, making it capable to learn and perform more complex tasks.
- The activation function must be
  - **differentiable** or the concept of updating weights (Gradient Descent and Backpropagation) fails, which is the core idea of deep learning.



  - "*There are more kind of activation function in use, we will discuss as required in upcoming classes*."

The – End.