CSL446
Neural Network and Deep Learning

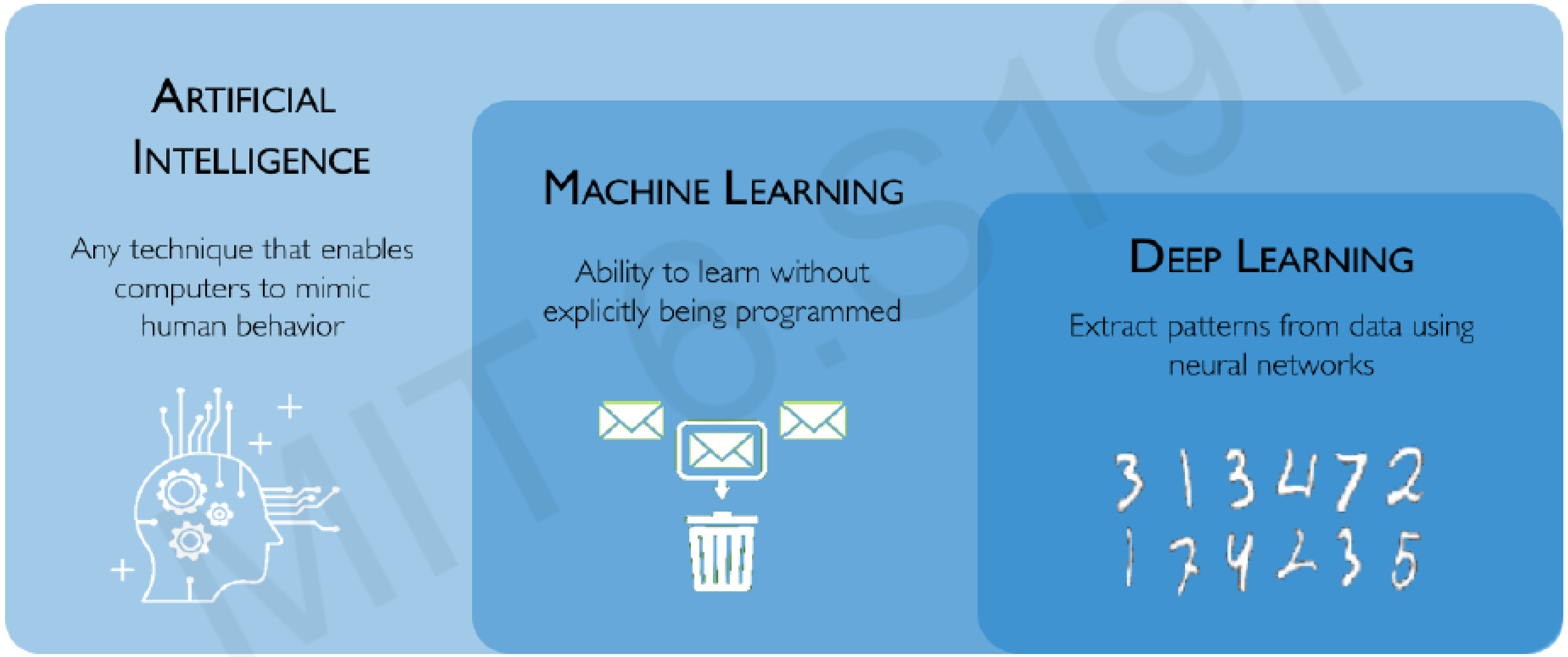
Module 1

Dr. Snehal B Shinde

Computer Science and Engineering

Indian Institute of Information Technology, Nagpur.

Types and Operations:



Artificial Intelligence

Any technique that enables computers to mimic human behavior

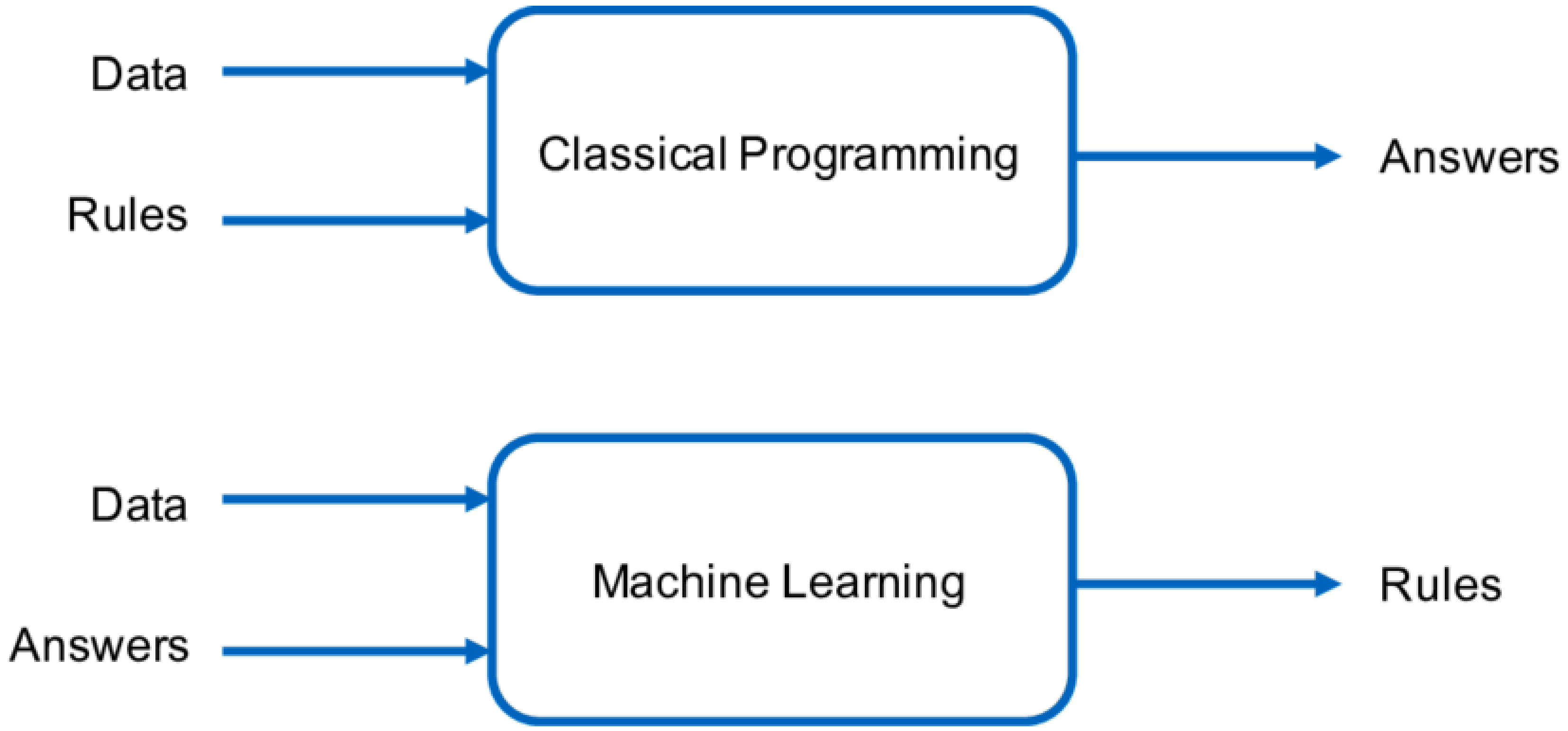
Machine Learning

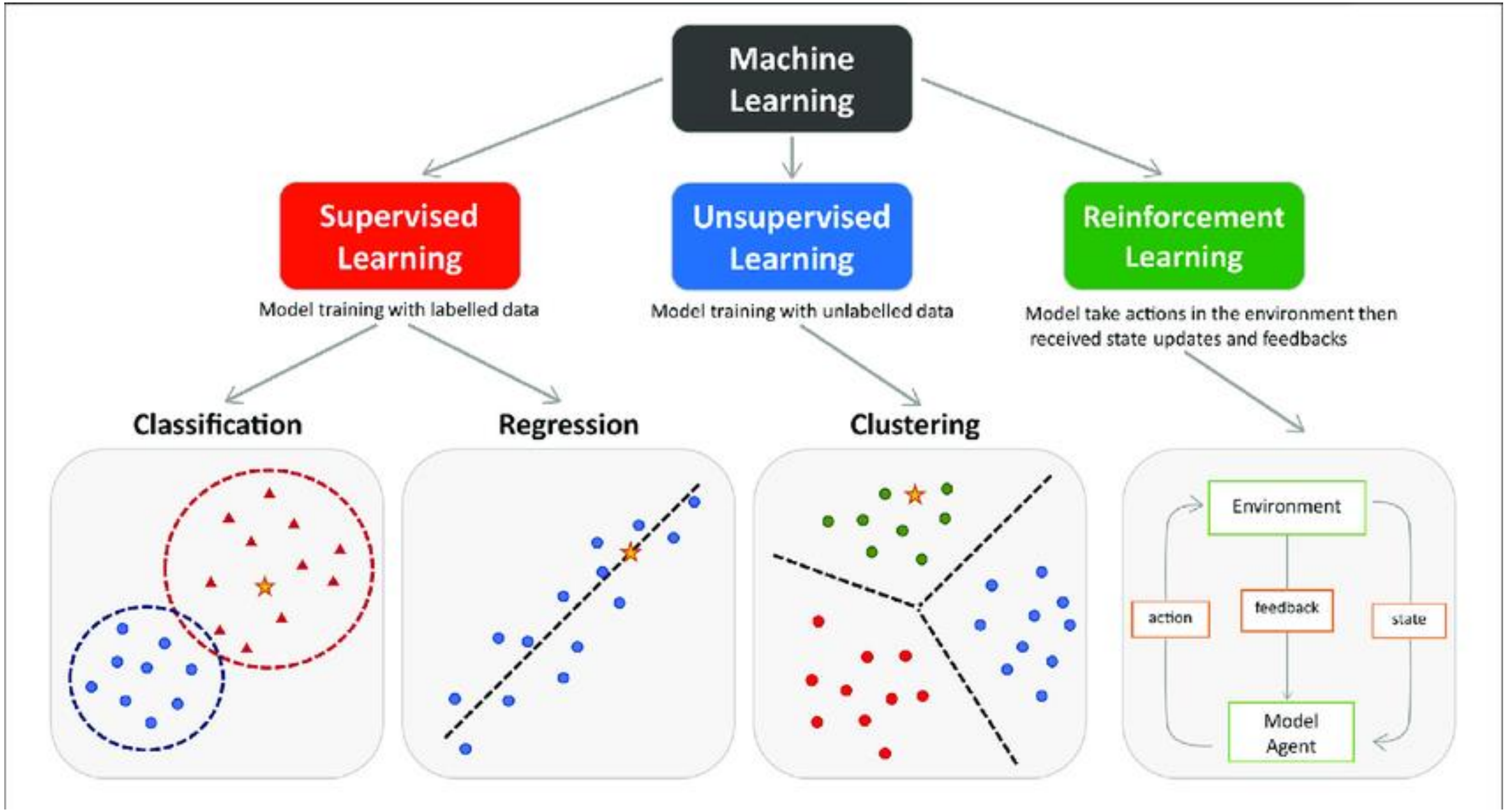
Ability to learn without explicitly being programmed

Deep Learning

Extract patterns from data using neural networks

Machine Learning




Machine Learning
Supervised Learning
Model training with labelled data
Unsupervised Learning
Model training with unlabelled data
Reinforcement Learning
Model take actions in the environment then received state updates and feedbacks
Classification
Regression
Clustering
Environment
action
feedback
state
Model Agent

Supervised

- **Supervised Learning is further divided into two categories:**
- **Regression:**

  In regression problems, the output is a continuous variable.

- **Classification:**

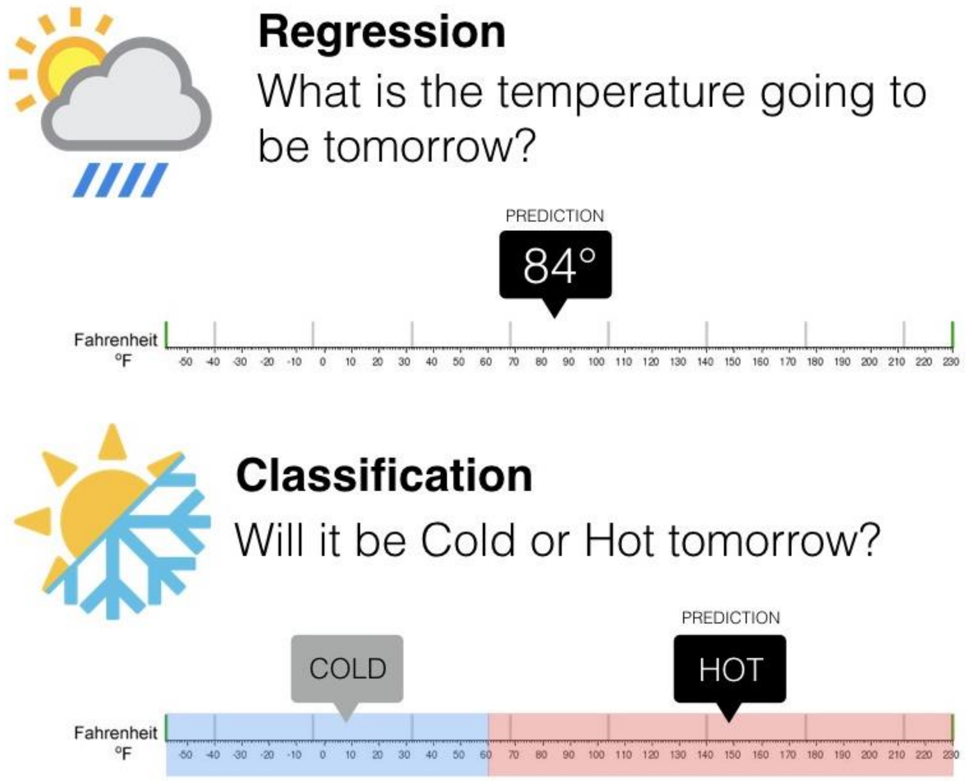
  There are two types of classifications in machine learning:

  - **Binary classification**: If the problem has only two possible classes, called a binary classifier. For example, cat or dog, Yes or No,
  - **Multi-class classification**: If the problem has more than two possible classes, it is a multi-class classifier.

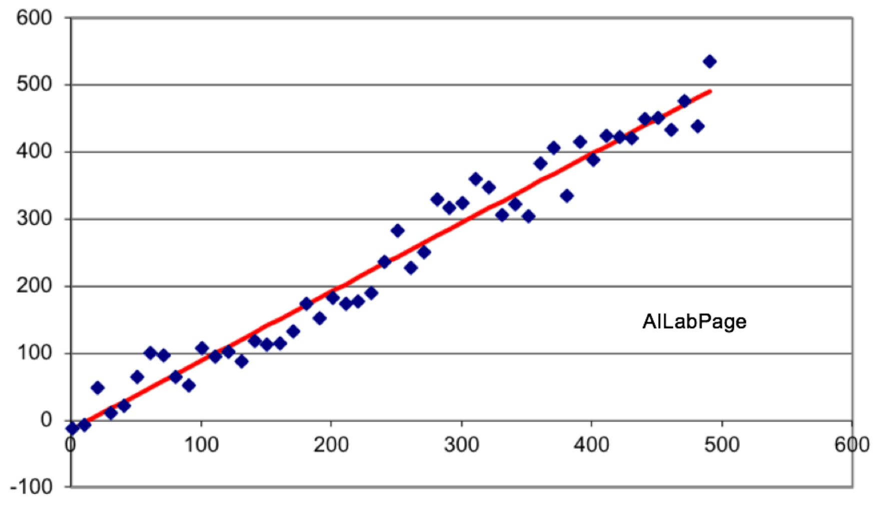
Classification vs. Regression

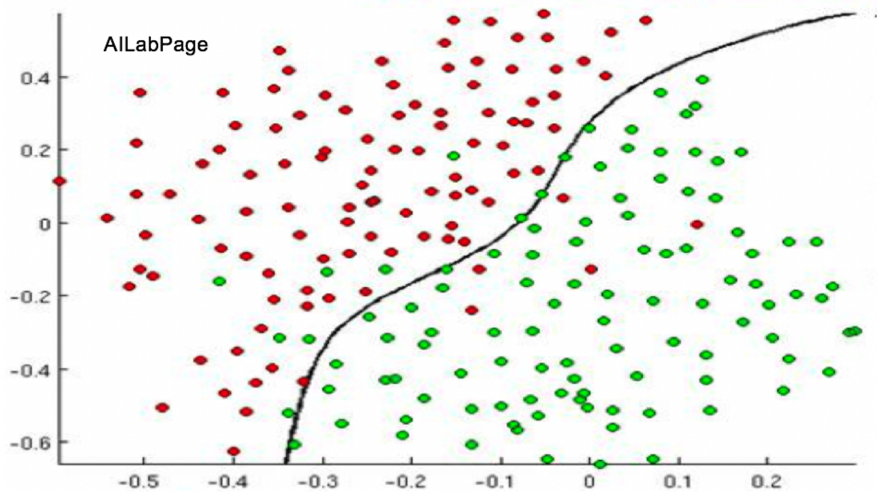
Regression algorithms predict continuous value from the provided input.
Predicting Stock Prices

In Classification, we try to find the decision boundary, which can divide the dataset into different classes.
Identification of spam emails, Speech Recognition, Identification of cancer cells



Classification vs. Regression



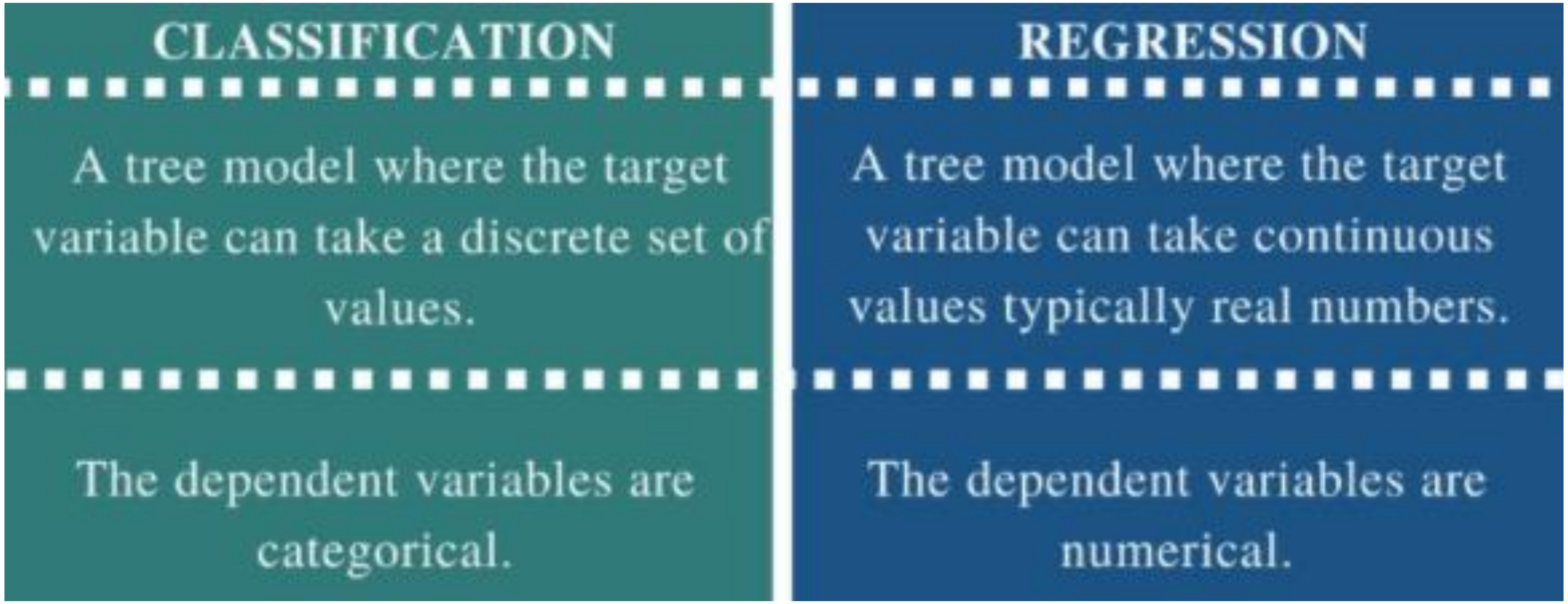


Regression
The system attempts to predict a value for an input based on past data.
Example – 1. Temperature for tomorrow

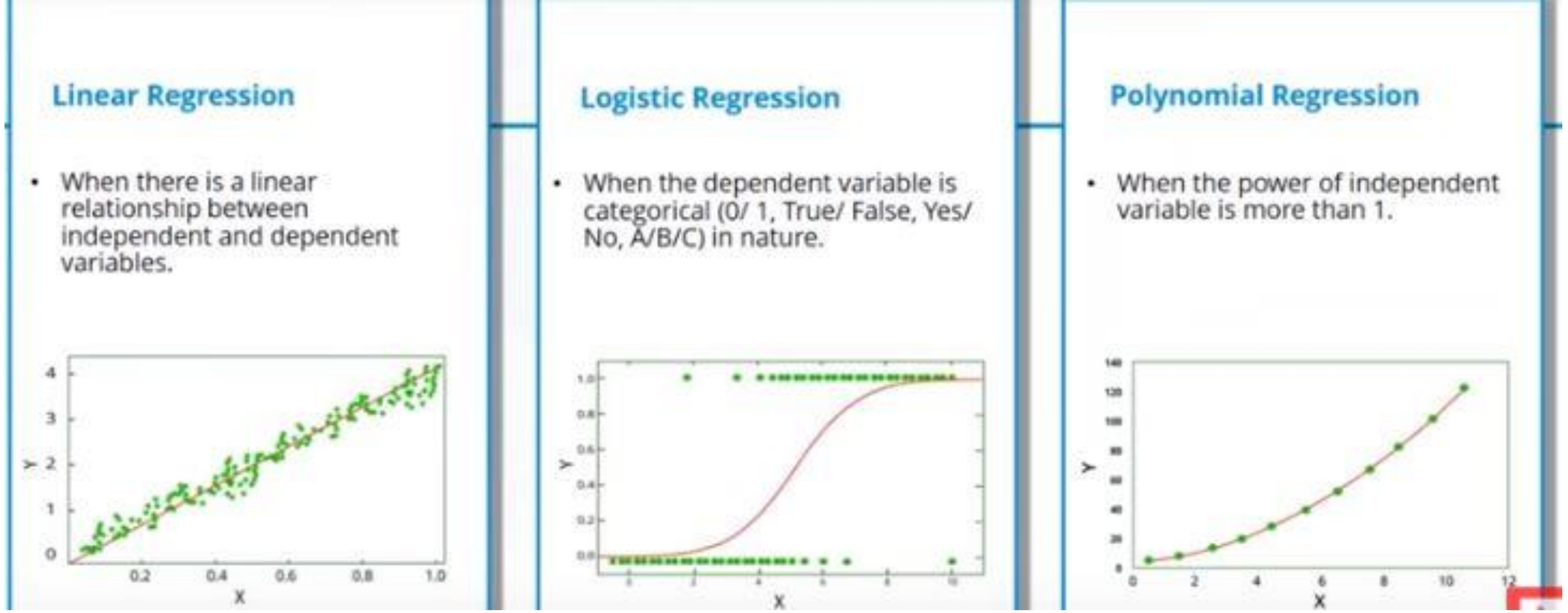
Classification
In classification, predictions are made by classifying them into different categories.
Example – 1. Type of cancer 2. Cancer Y/N

AILabPage

Classification vs. Regression



CLASSIFICATION	REGRESSION
A tree model where the target variable can take a discrete set of values.	A tree model where the target variable can take continuous values typically real numbers.
The dependent variables are categorical.	The dependent variables are numerical.



Linear Regression

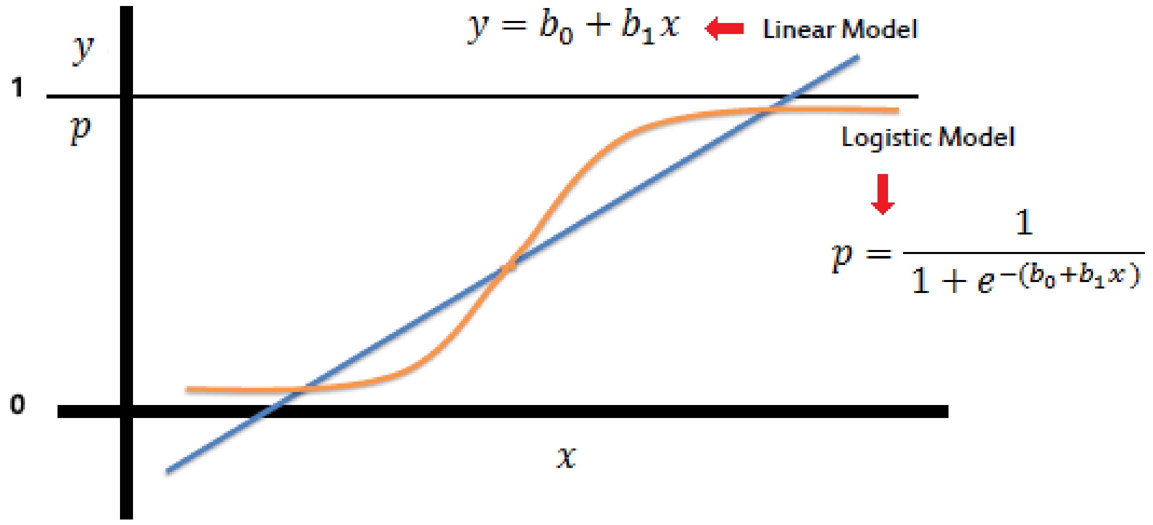
- When there is a linear relationship between independent and dependent variables.

Logistic Regression

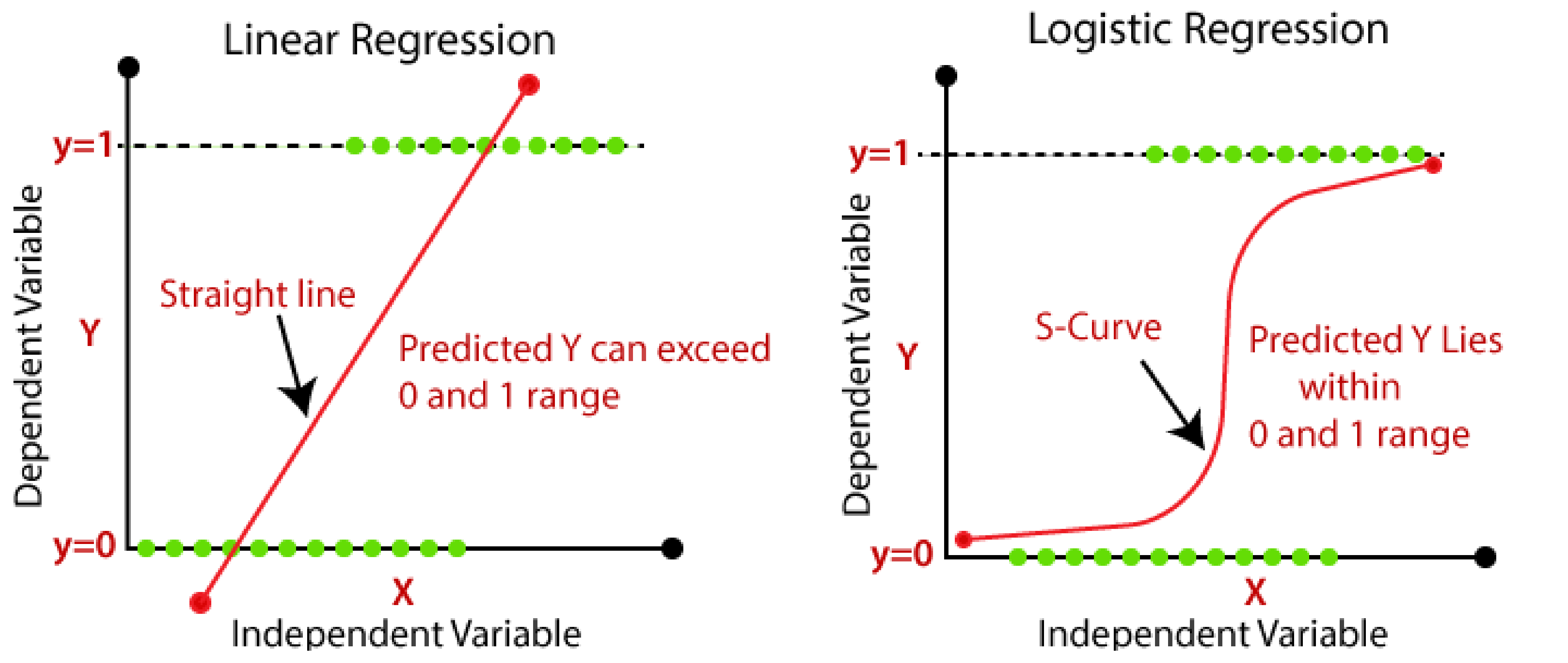
- When the dependent variable is categorical (0/ 1, True/ False, Yes/ No, A/B/C) in nature.

Polynomial Regression

- When the power of independent variable is more than 1.



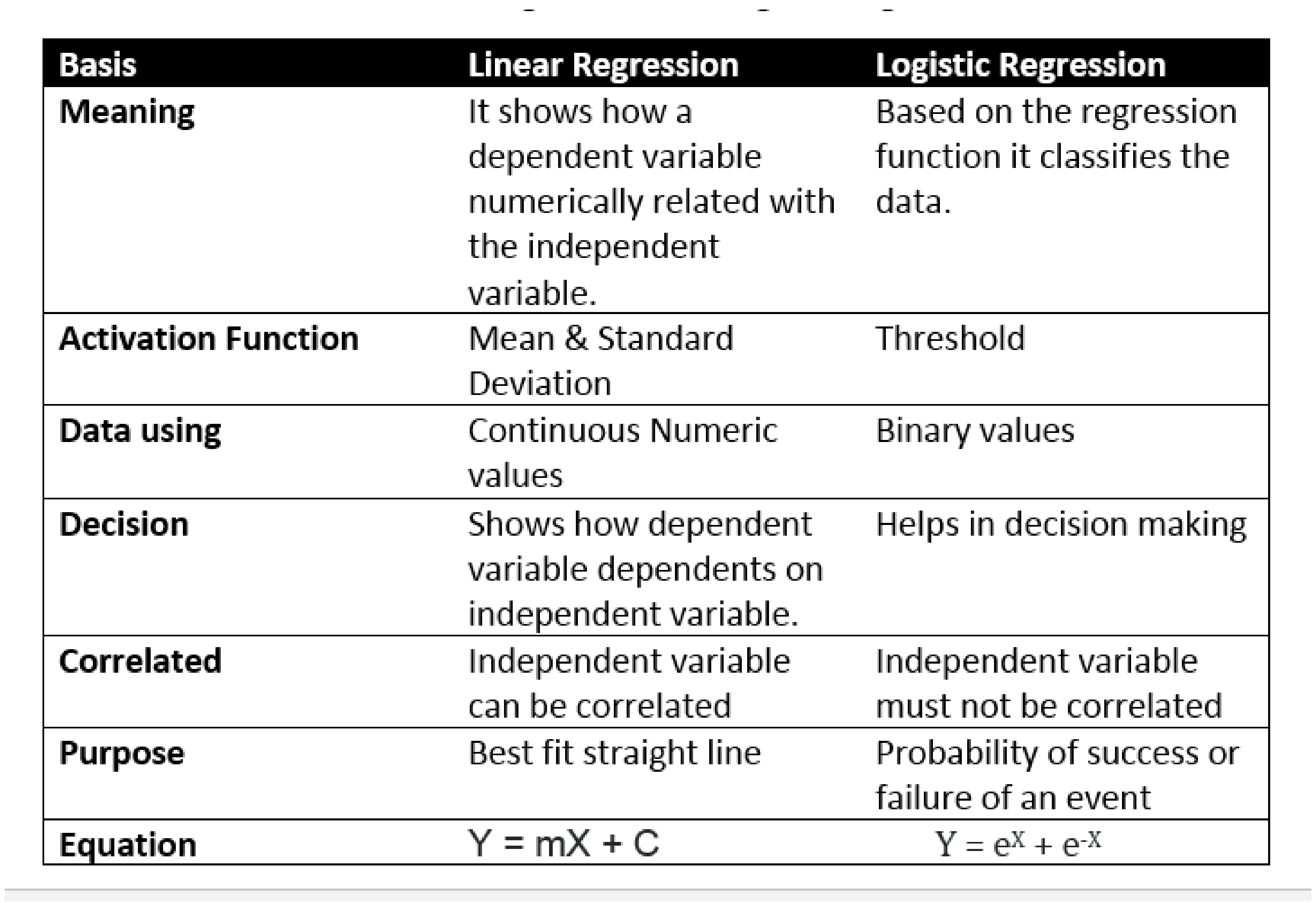
Linear Regression vs. Logistic Regression



Logistic Regression

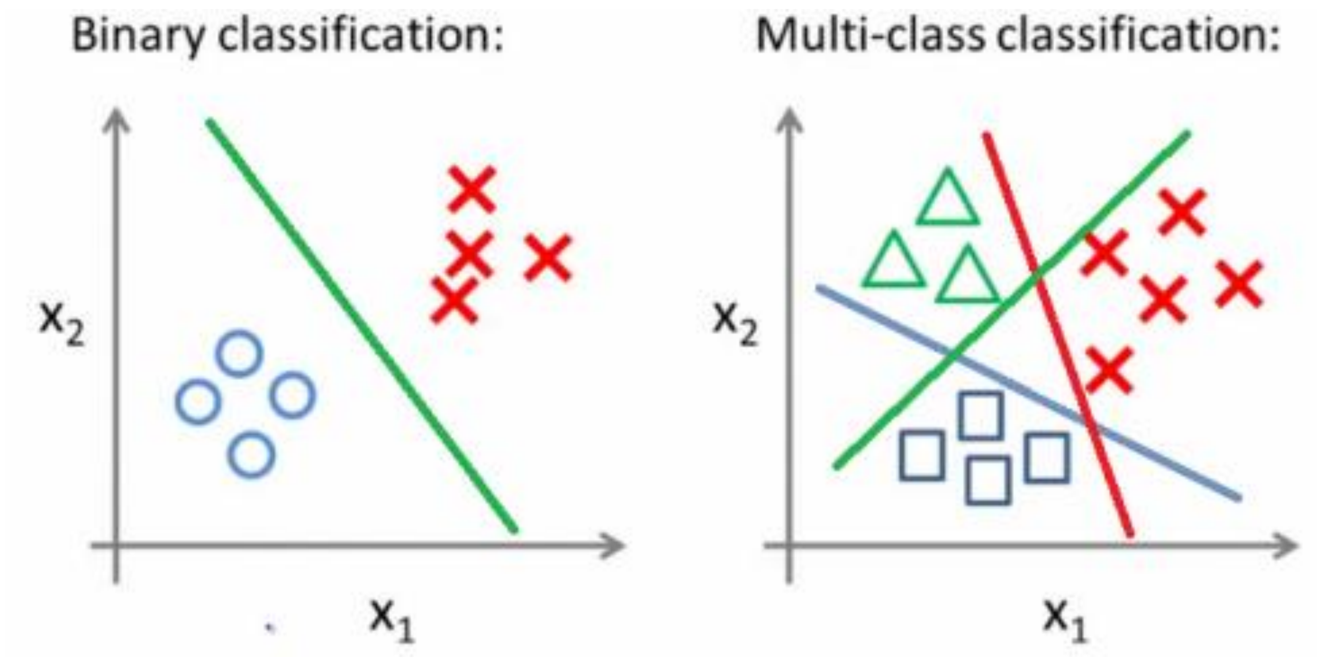
- For example, suppose the classification threshold is 0.82
- Imagine an example that produces a raw prediction 2.6. The sigmoid of 2.6 is 0.93. Since 0.93 is greater than 0.82, the system classifies this example as the positive class.
- Imagine a different example that produces a raw prediction of 1.3. The sigmoid of 1.3 is 0.79. Since 0.79 is less than 0.82, the system classifies that example as the negative class.
- Increase threshold if we want to reduce FP
- Decrease threshold if we want to reduce FN

Linear Regression vs. Logistic Regression



Basis	Linear Regression	Logistic Regression
Meaning	It shows how a dependent variable numerically related with the independent variable.	Based on the regression function it classifies the data.
Activation Function	Mean & Standard Deviation	Threshold
Data using	Continuous Numeric values	Binary values
Decision	Shows how dependent variable dependents on independent variable.	Helps in decision making
Correlated	Independent variable can be correlated	Independent variable must not be correlated
Purpose	Best fit straight line	Probability of success or failure of an event
Equation	$Y = mX + C$	$Y = e^{X} + e^{-X}$

Binary vs. Multiclass classification



- Multi-class classification provides more than two outcomes of the model
- Handwritten digit recognition (0-9 digits), email categorization (spam, primary, social, promotions).

Performance Measures of Regression

- Mean Squared Error (MSE):

Penalizes large errors more heavily than small errors.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- Mean Absolute Error (MAE):

Sensitive to outliers, but less so than MSE.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

- Root Mean Squared Error (RMSE):

Preferred when you want to emphasize larger errors more in the evaluation metric.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Performance Measures of Regression

R-Squared:

R2 is the proportion of variance in the dependent variable (y) explained by the independent variables in the model.

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}}$$

Where:

- $\text{SS}_{\text{res}} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$ (Residual Sum of Squares)
- $\text{SS}_{\text{tot}} = \sum_{i=1}^{n} (y_i - \bar{y})^2$ (Total Sum of Squares)

Adjusted R-squared:

- Adjusted R^2modifies R^2 to account for the number of predictors in the model.
- It penalizes the inclusion of irrelevant predictors that do not improve the model.

$$R^2_{\text{adj}} = 1 - \left(\frac{1 - R^2}{n - k - 1} \right) (n - 1)$$

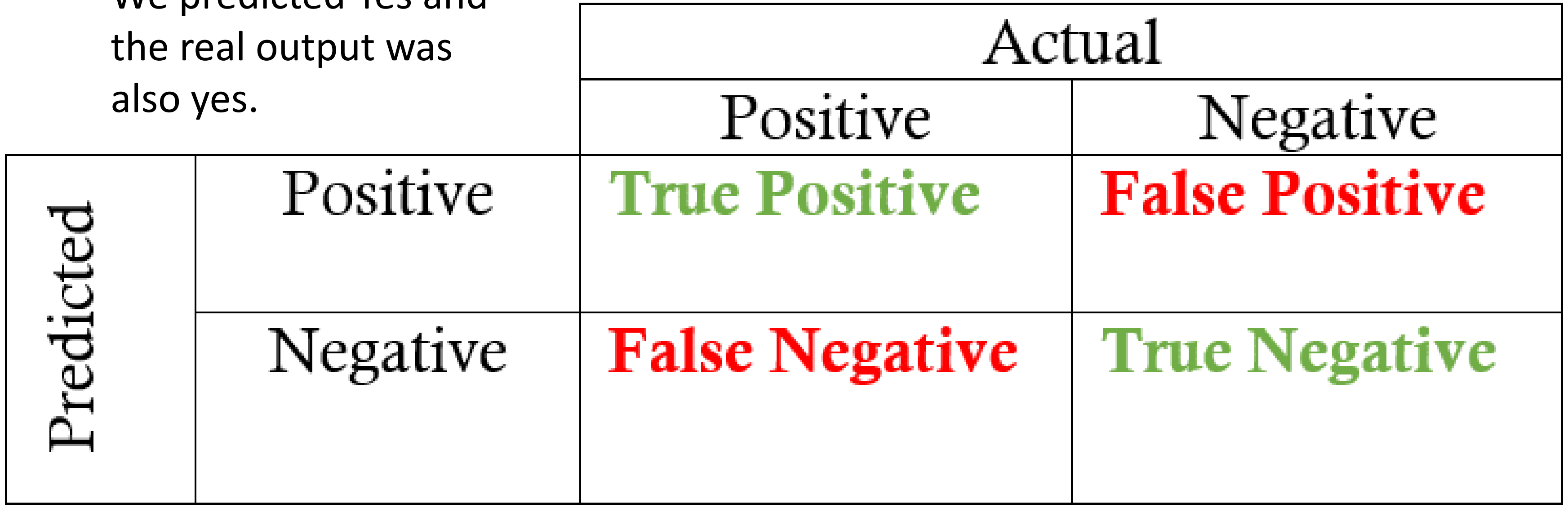
Performance Measures of Classification

- Accuracy
- Recall/Sensitivity
- Precision
- Specificity
- F1-score
- AUC-ROC

Confusion Matrix

We predicted Yes and the real output was also yes.

we predicted Yes but it was actually No.



		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

we predicted No but it was actually Yes.

we predicted No and the real output was also No.

Accuracy

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Number Of Examples}}$$

In **binary classification** (#binary_classification), accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Number Of Examples}}$$

Recall/Sensitivity/TPR

- A metric for **classification models** that answers the following question.
- Out of all the possible positive labels, how many did the model correctly identify?

$$\text{Recall} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Negative}}$$

- Important when false negatives are costly (e.g., detecting cancer or fraud).

Precision

- Precision is useful when we want to reduce the number of False Positives.
- Ex. E-mail spam

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

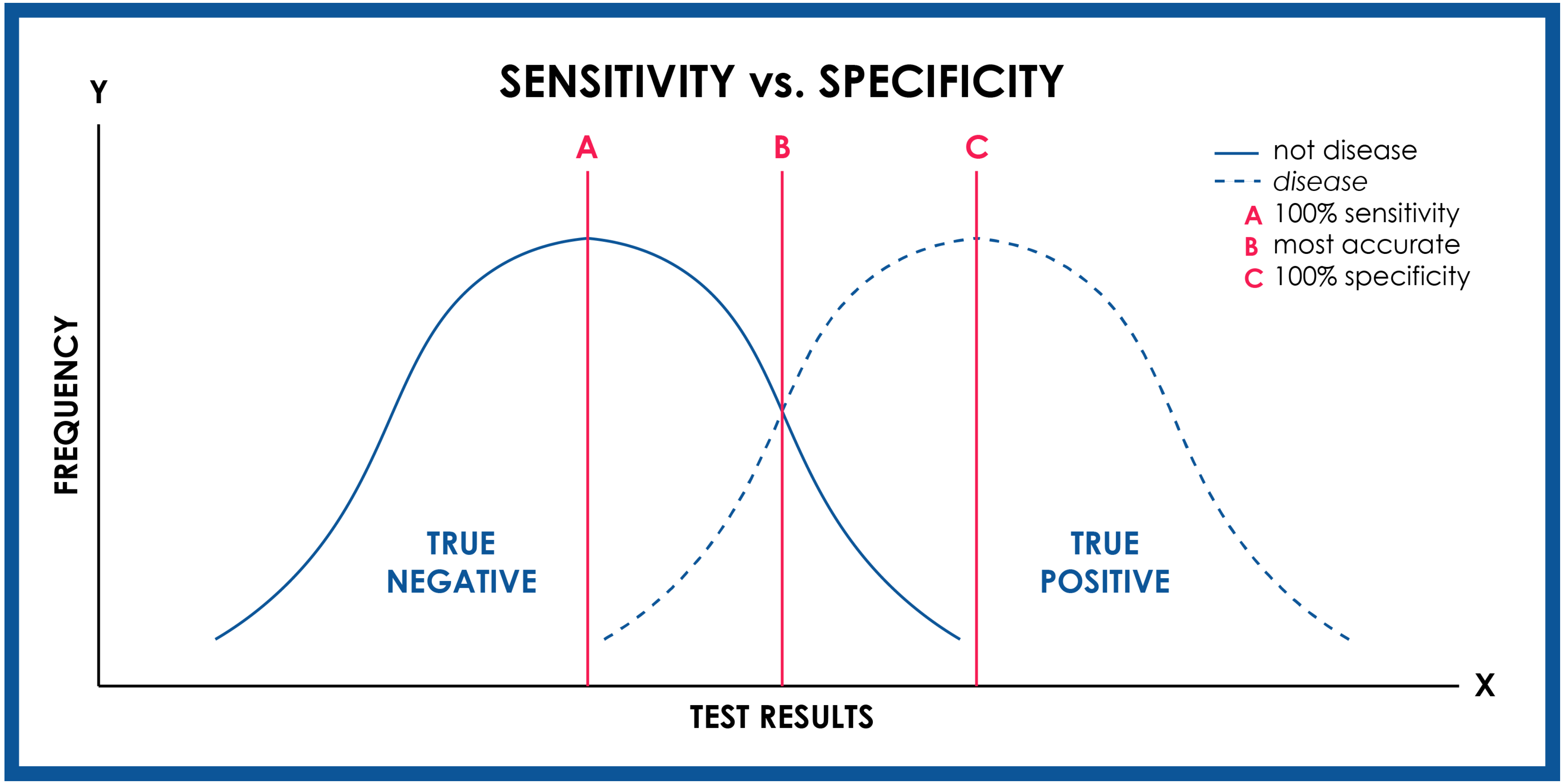
- Important when false positives are costly (e.g., spam detection).

Specificity / True Negative Rate

- Specificity is defined as the ratio of True negatives and True negatives + False positives.
- We want the value of specificity to be high.
- Its value lies between [0,1].

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$

Specificity / True Negative Rate



F1-Score

- F1-score is a metric that combines both Precision and Recall
- harmonic mean of precision and recall
- Its value lies between [0,1]
- More the value better the F1-score.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

TPR and FPR

$$\text{True Positive Rate (TPR)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{False Positive Rate (FPR)} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

mAP

- In computer vision, mAP is a popular evaluation metric used for object detection (i.e. localization and classification).

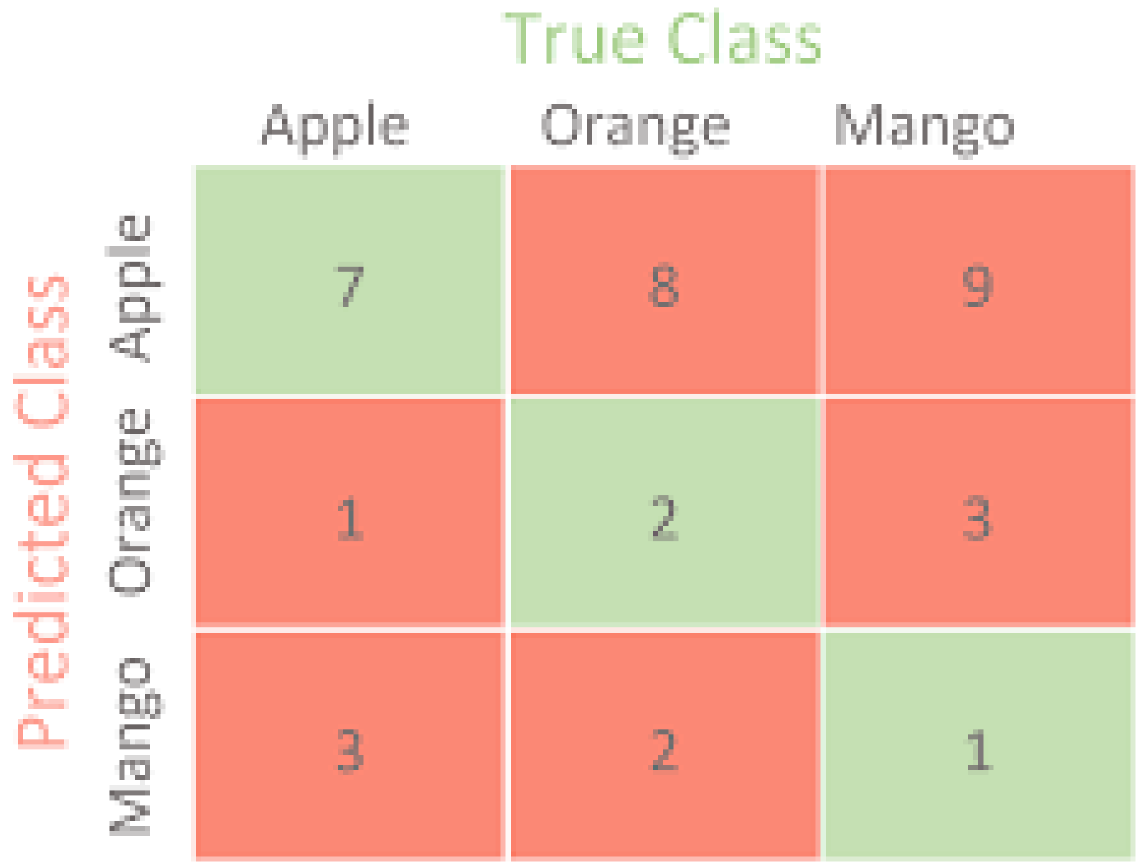
Confusion Matrix

- An NxN table that summarizes how successful a **classification model's** predictions were;

- The correlation between the label and the model's classification.

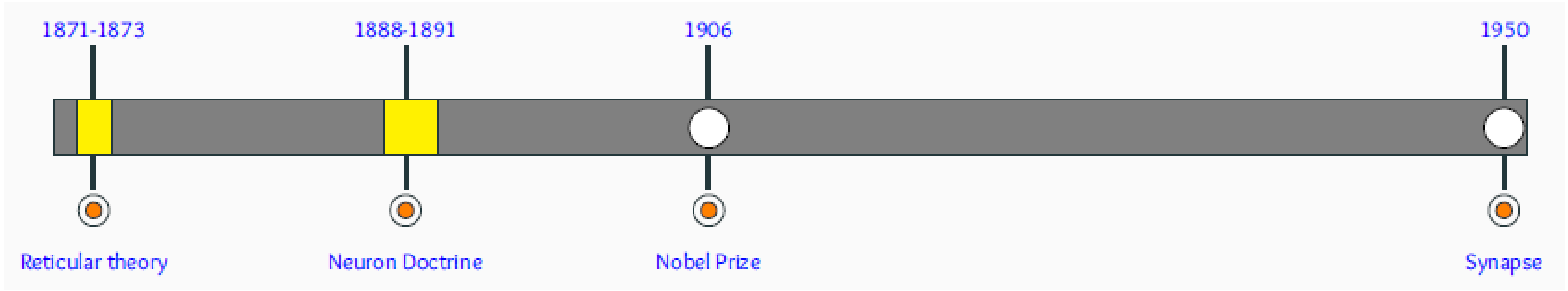
	Tumor (predicted)	**Non-Tumor (predicted)**
Tumor (actual)	18	1
Non-Tumor (actual)	6	452

Confusion matrix for MCC



		True Class		
		Apple	Orange	Mango
Predicted Class	Apple	7	8	9
	Orange	1	2	3
	Mango	3	2	1

Biological Neurons



Reticular Theory

Joseph von Gerlach proposed that the nervous system is a single continuous network as opposed to a network of many discrete cells!

Neuron Doctrine

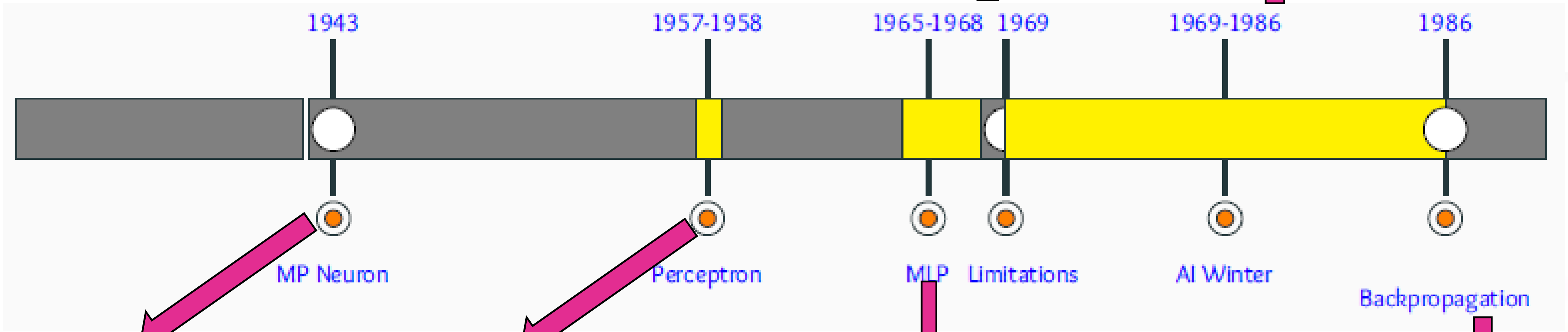
Santiago Ramón y Cajal used Golgi's technique to study the nervous system and proposed that it is actually made up of discrete individual cells forming a network (as opposed to a single continuous network)

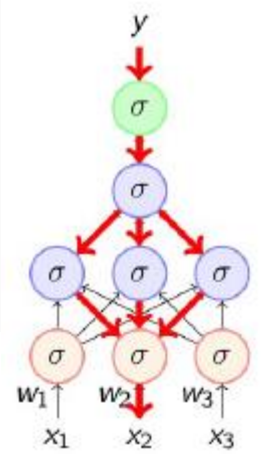
Both Golgi (reticular theory) and Cajal (neuron doctrine) were jointly awarded the 1906 Nobel Prize for Physiology or Medicine, that resulted in lasting conflicting ideas and controversies between the two scientists.

In 1950s electron microscopy finally confirmed the neuron doctrine by unambiguously demonstrating that nerve cells were individual cells interconnected through synapses (a network of many individual neurons).

Perceptron Limitations : Minsky and Papert highlighted the limitations of the perceptron in their book *Perceptrons*, particularly its inability to solve non-linear problems like XOR.

Almost lead to the abandonment of connectionist AI Winter





McCulloch Pitts Neuron

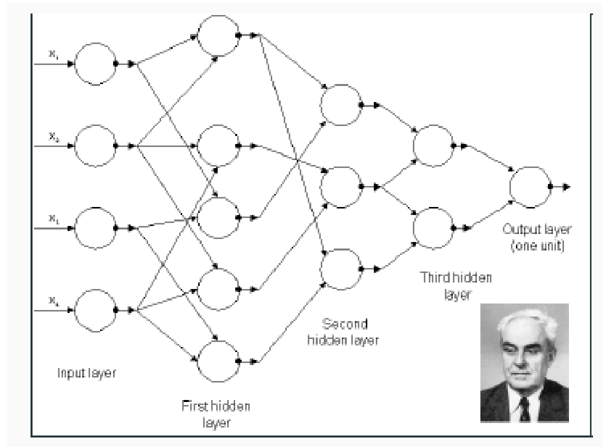
McCulloch (neuroscientist) and Pitts (logician) proposed a highly simplified model of the neuron (1943)

perceptron

the perceptron may eventually be able to learn, make decisions, and translate languages" -Frank Rosenblatt

First generation Multilayer Perceptrons

Ivakhnenko et. al



The backpropagation algorithm, rediscovered by Rumelhart, Hinton, and Williams in 1986, became a key method for training multi-layer neural networks, overcoming earlier limitations.

Breakthroughs in Deep Learning (2010–2015)

2012: CNN

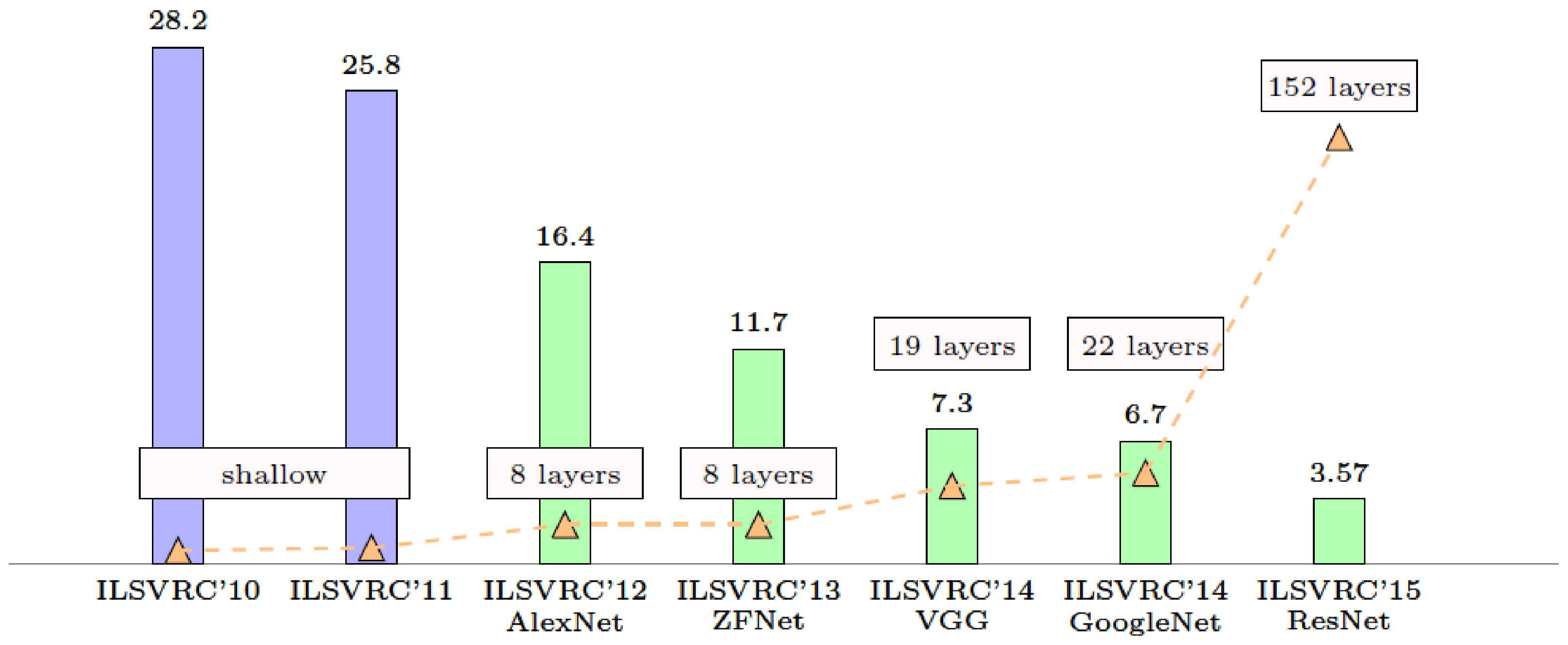
Convolutional Neural Networks (CNNs) were first introduced in the **1980s**.
LeCun's **LeNet-5** (1998) is one of the earliest and most famous CNN architectures.

AlexNet and ImageNet

- AlexNet marked a revolutionary moment in deep learning when it won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. This achievement highlighted the power of deep convolutional neural networks (CNNs) for image classification tasks.
- **Layers:** AlexNet consisted of 8 layers: 5 convolutional layers and 3 fully connected layers.
- **ReLU Activation:** ReLU (Rectified Linear Unit) was used instead of traditional activation functions (like sigmoid or tanh), solving the vanishing gradient problem and speeding up training.
- **Dropout Regularization:** Introduced dropout to prevent overfitting during training.

CNN Architectures

Network	Error	Layers
AlexNet[	16.0%	8
ZFNet	11.2%	8
VGGNet	7.3%	19
GoogLeNet	6.7%	22



Object Detection and Recognition (Post-2012)

- **R-CNN (Regions with CNN Features, 2014):**
  - Proposed by Ross Girshick, uses Selective Search and SVM Computationally expensive .

- **Fast R-CNN and Faster R-CNN (2015):**
  - Improvements over R-CNN with better speed and accuracy.
  - Faster R-CNN introduced the Region Proposal Network (RPN), which replaced traditional methods for generating region proposals, making object detection more efficient.

- **YOLO (You Only Look Once, 2016):**
  - Developed by Joseph Redmon, YOLO framed object detection as a single regression problem.
  - Achieved real-time detection speeds with good accuracy by predicting bounding boxes and class probabilities in one pass.

Applications:
Autonomous Vehicles: Object detection is critical for identifying pedestrians, vehicles, and traffic signs.
Healthcare: Detecting tumors or abnormalities in medical imaging.
Surveillance: Recognizing individuals or activities in security systems.
Augmented Reality (AR): Enabling applications to interact with real-world objects.

Breakthroughs in Deep Learning (2010–2015)- **Recurrent Neural Networks (RNNs)**

- RNNs became a critical innovation for handling sequential data, such as time series, speech, and text, where the order of data points matters.
- RNNs were prone to the **vanishing gradient problem**, which made it difficult to learn long-term dependencies.
- Long Short-Term Memory (LSTM) networks, introduced by Hochreiter and Schmidhuber, solved the vanishing gradient problem by introducing memory cells and gates (input, forget, and output gates) to control the flow of information.
- LSTMs demonstrated the importance of memory and context in neural networks, paving the way for more advanced models like **GRUs (Gated Recurrent Units)** and Transformers.

Generative Models:

- Generative Models emerged as a powerful class of deep learning techniques and Introduced by Ian Goodfellow in 2014, focusing on generating new data samples that resemble the input data distribution.
- Autoencoders and Variational Autoencoders (VAEs) emerged for unsupervised learning tasks.

- **Autoencoders**

Autoencoders compress (encode) and reconstruct (decode) input data to learn efficient representations.

1. Image Denoising: Removes noise from images by learning clean patterns.
2. Anomaly Detection: Identifies outliers by measuring reconstruction errors.

Variational Autoencoders (VAEs)

Learn a probabilistic distribution of the latent space for generative tasks. Specifically designed for generating new, realistic samples.

1. Synthetic Data Generation: Creates realistic data for training.
2. Medical Imaging: Generates new medical scans for research and diagnostics.

Generative Adversarial Networks (GANs)

GANs, introduced by Ian Goodfellow in 2014, revolutionized generative modeling by creating realistic data through adversarial training.

Architecture:

Generator: Synthesizes fake data.
Discriminator: Differentiates between real and fake data.
Both networks improve iteratively, resulting in high-quality data synthesis.

Applications:

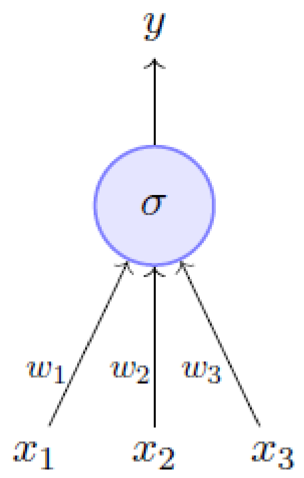
- **Image Synthesis:** Creating realistic images (e.g., DeepFake technology).
- **Style Transfer:** Adapting one image's style to another.
- **3D Modeling:** Generating realistic 3D assets for gaming and design.

GANs transformed generative modeling, enabling applications in creative industries, simulation, and more.

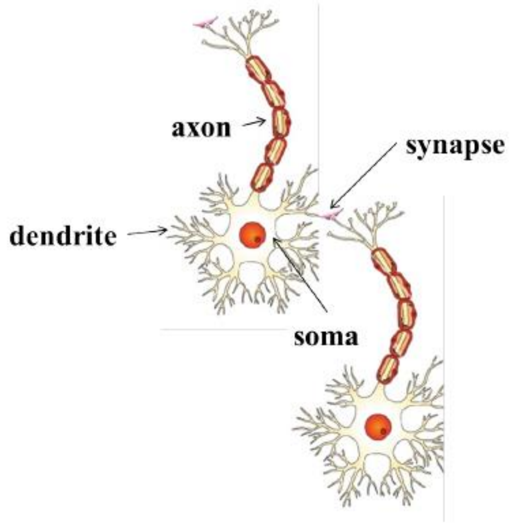
Year	Milestone	Key Contributions	Impact
1943	McCulloch-Pitts Model	First computational model of artificial neurons.	Laid the theoretical foundation for neural networks.
1950s	Perceptron	Introduced by Frank Rosenblatt; capable of learning weights and biases.	First practical implementation of artificial neural networks.
1980s	Backpropagation Algorithm	Rediscovered by Rumelhart, Hinton, and Williams; enabled multi-layer networks.	Solved key training issues for deeper networks.
1990s	LSTM (Long Short-Term Memory)	Introduced by Hochreiter and Schmidhuber; solved vanishing gradient problem in RNNs.	Enabled sequential data processing for tasks like speech and text.
2012	AlexNet	Deep CNN won the ImageNet competition; used ReLU, dropout, and GPUs for training.	Revolutionized computer vision and popularized deep learning.
2014	GANs (Generative Adversarial Networks)	Introduced by Ian Goodfellow; adversarial training for generating realistic samples.	Enabled applications in image synthesis, style transfer, and more.
2015	ResNet	Introduced skip connections to solve vanishing gradient problem in very deep networks.	Enabled training of ultra-deep networks effectively.
2016	YOLO (You Only Look Once)	Real-time object detection in a single pass.	Made object detection fast and efficient for real-world applications.
2017	Transformers	Introduced in 'Attention Is All You Need'; revolutionized NLP.	Basis for models like BERT, GPT, and Vision Transformers (ViT).
2020	GPT-3	A massive Transformer-based language model.	Achieved state-of-the-art performance in natural language understanding and generation.
2020s	Multimodal Models (DALL-E, CLIP)	Combined vision and language tasks.	Enabled applications like text-to-image generation and enhanced AI understanding.

Biological Neurons

- The most fundamental unit of a deep neural network is called an artificial neuron
- Why is it called a neuron ?
- Where does the inspiration come from ?
- The inspiration comes from biology (more specically, from the brain)
- biological neurons = neural cells = neural processing units



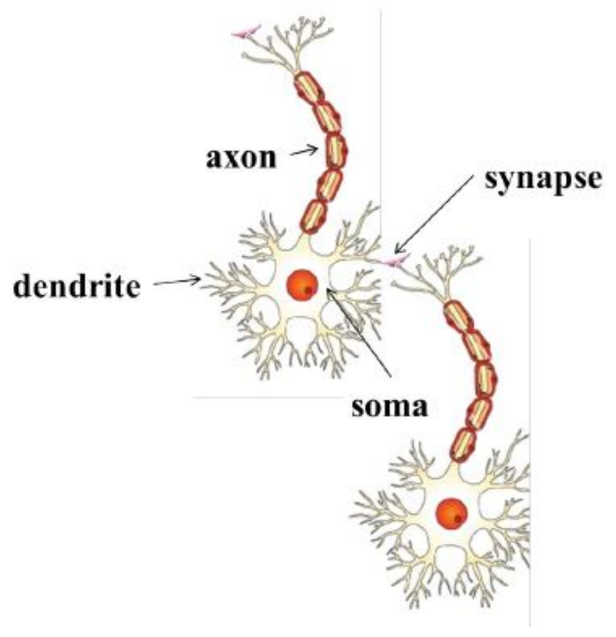
Artificial Neuron



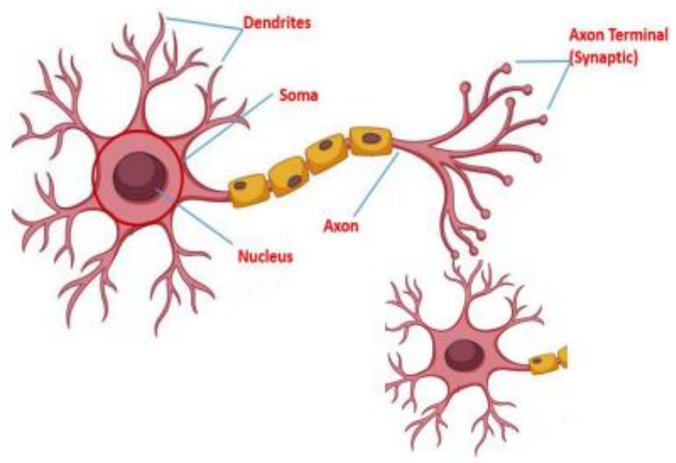
Biological Neurons*

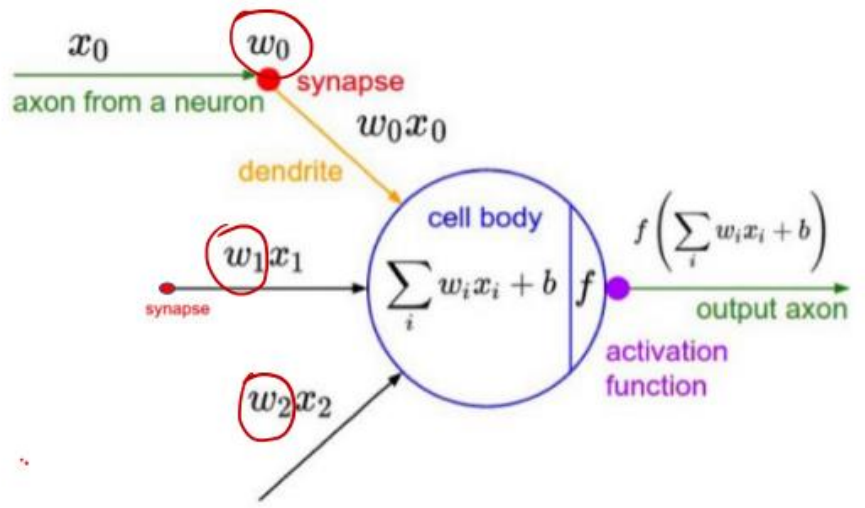
Biological Neurons

- dendrite: receives signals from other neurons
- synapse: point of connection to other neurons
- soma: processes the information
- axon: transmits the output of this neuron



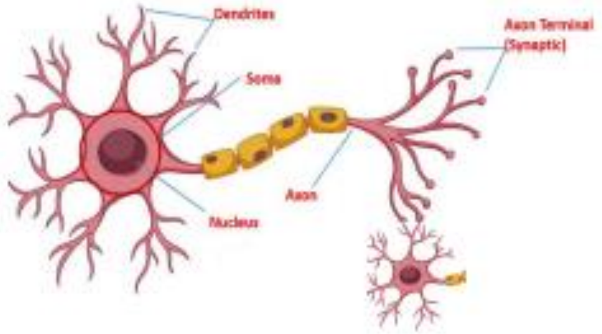
Biological Neurons*

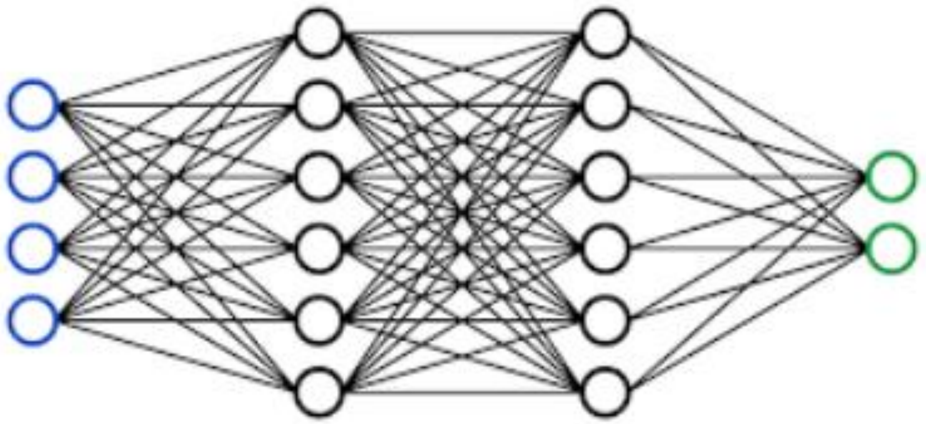




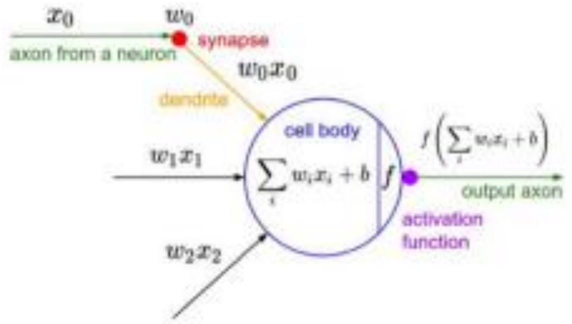
Biological Neurons

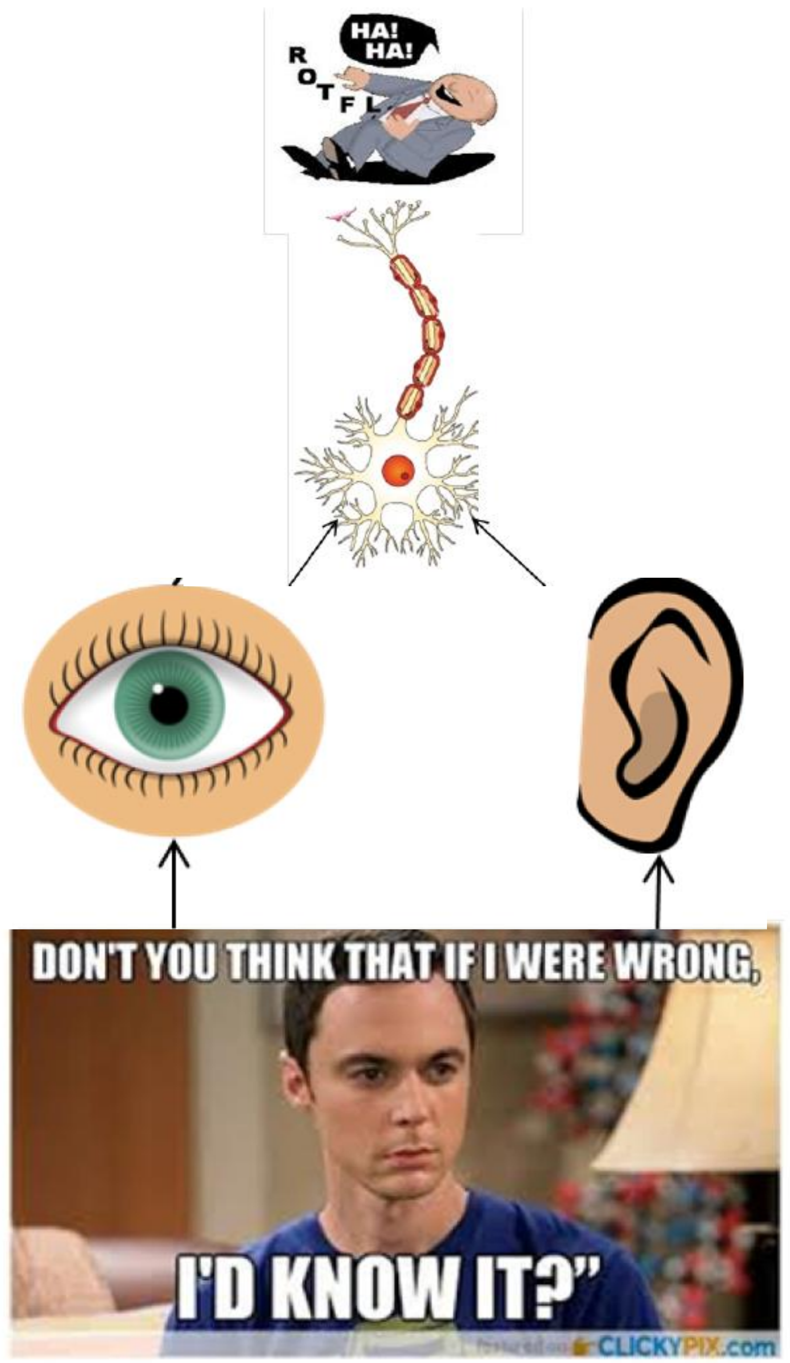
Network of neurons



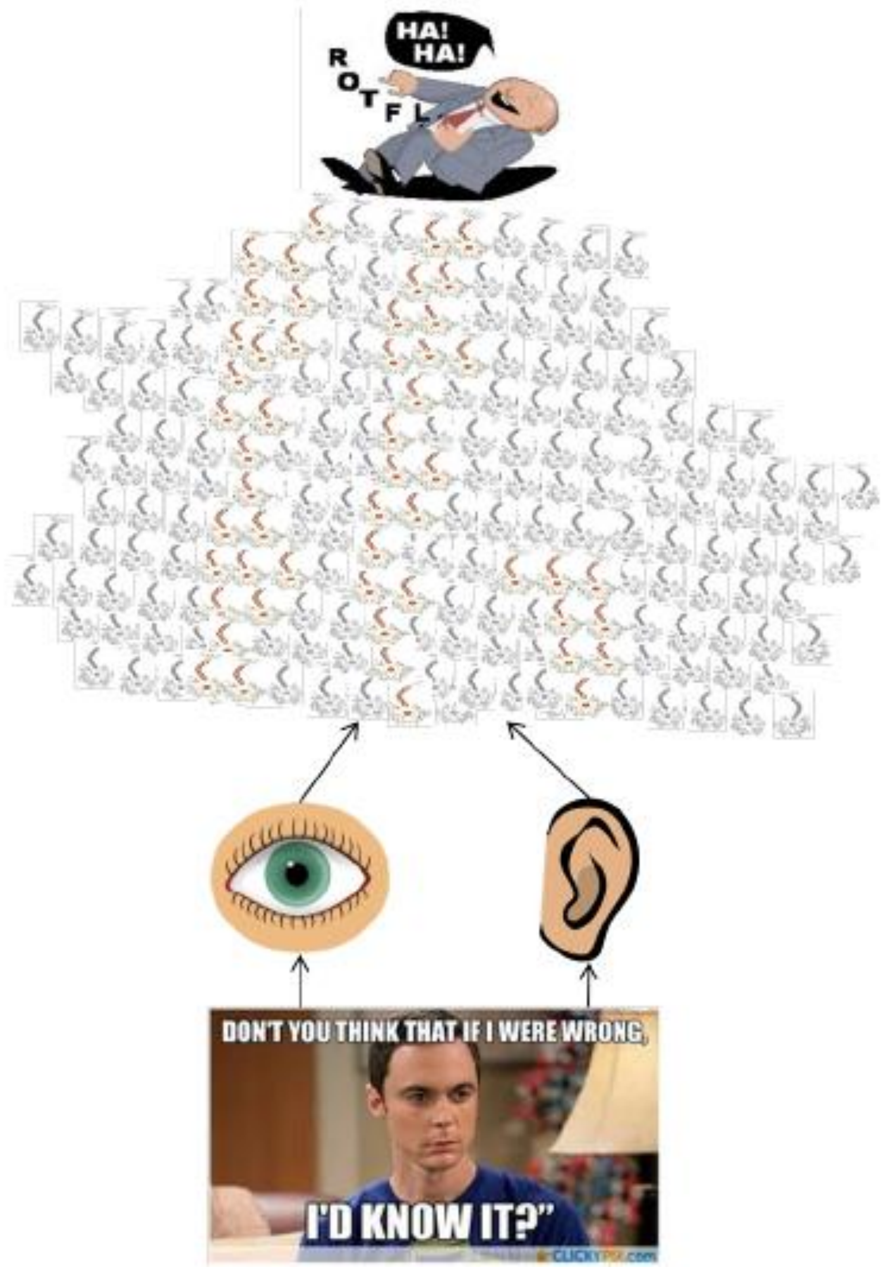


Network of neurons

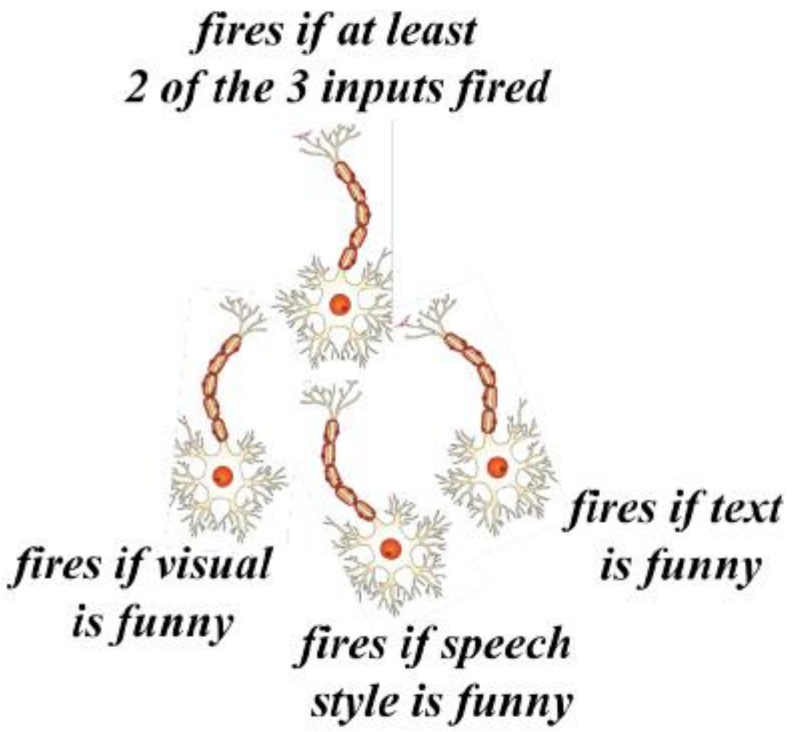




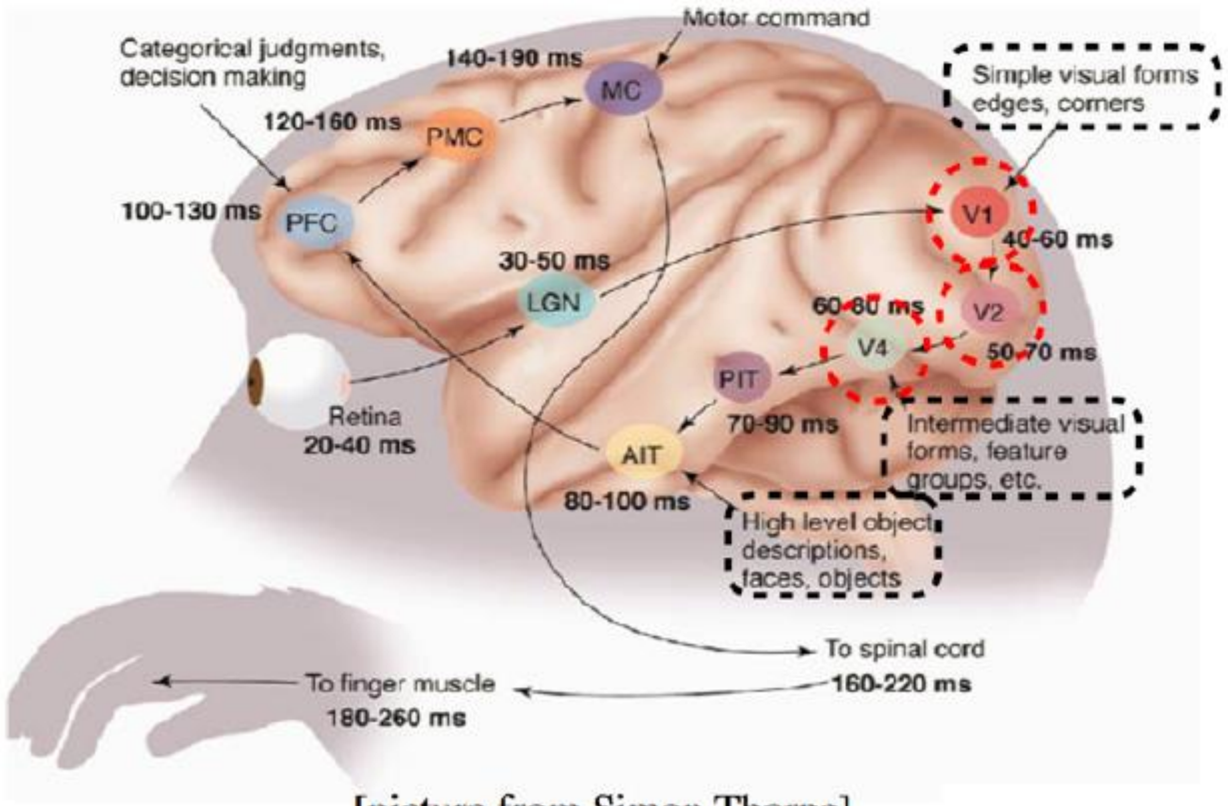
- Let us see a very cartoonish illustration of how a neuron works
- Our sense organs interact with the outside world
- They relay information to the neurons
- The neurons (may) get activated and produces a response (laughter in this case)



- It is not just a single neuron which does all this.
- There is a massively parallel interconnected network of neurons.
- The sense organs relay information to the lowest layer of neurons.
- Some of these neurons may fire (in red) in response to this information and in turn relay information to other neurons they are connected to.
- These neurons may also fire (again, in red) and the process continues eventually resulting in a response (laughter in this case)
- An average human brain has around 10^{11} (100billion) neurons!


fires if at least 2 of the 3 inputs fired
fires if visual is funny
fires if speech style is funny
fires if text is funny

- This massively parallel network also ensures that there is division of work
- Each neuron may perform a certain role or respond to a certain stimulus

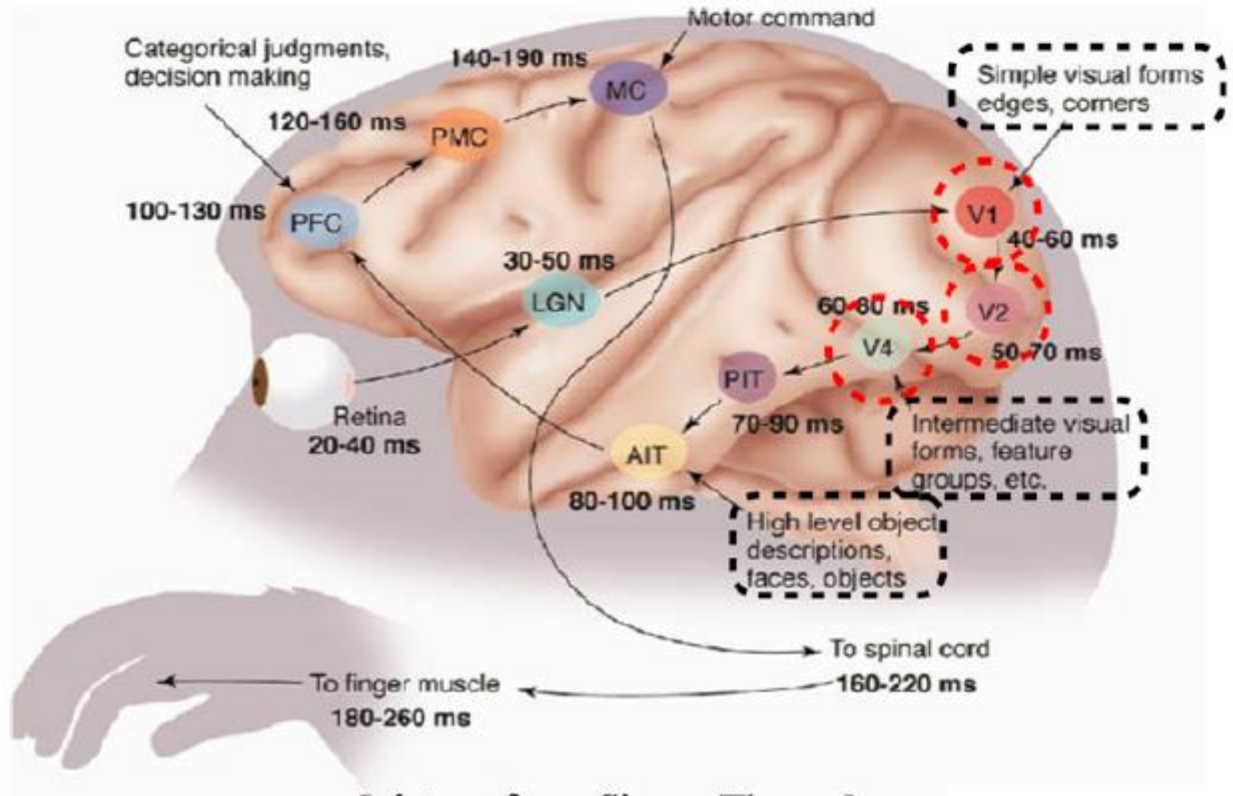


[picture from Simon Thorpe]

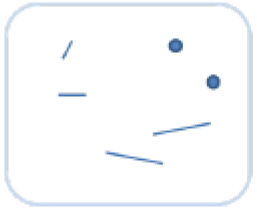
- The neurons in the brain are arranged in a hierarchy. We illustrate this with the help of visual cortex (part of the brain) which deals with processing visual information

- Starting from the retina, the information is relayed to several layers (follow the arrows)

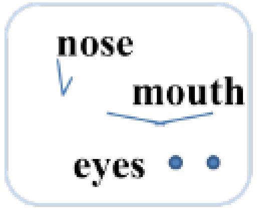
- We observe that the layers V 1, V 2 to AIT form a hierarchy (from identifying simple visual forms to high level objects)



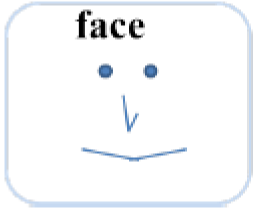
[picture from Simon Thorpe]



Layer 1: detect edges & corners

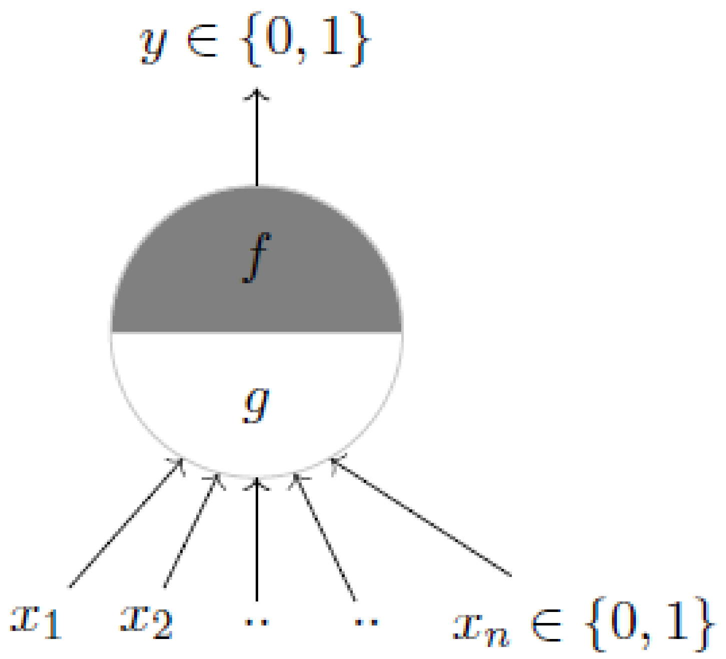


Layer 2: form feature groups



Layer 3: detect high level objects, faces, etc.

MP Neurons

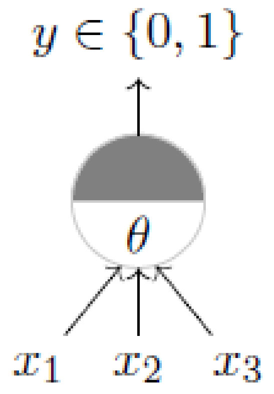


- McCulloch (neuroscientist) and Pitts (logician) proposed a highly simplified computational model of the neuron (1943)
- **g** aggregates the inputs and the function f takes a decision based on this aggregation

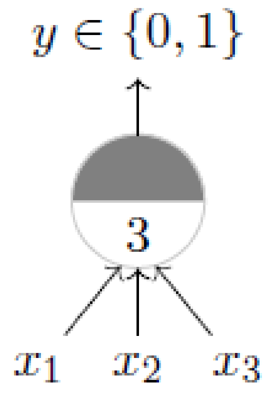
$$g(x_1, x_2, ..., x_n) = g(\mathrm{x}) = \sum_{i=1}^{n} x_i$$

$$\begin{aligned} y = f(g(\mathrm{x})) &= 1 \quad if \quad g(\mathrm{x}) \geq \theta \\ &= 0 \quad if \quad g(\mathrm{x}) < \theta \end{aligned}$$

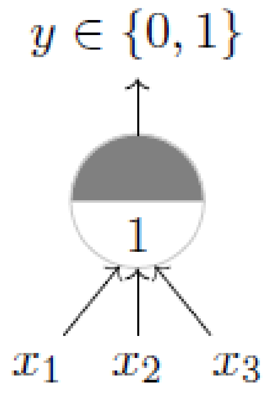
MP Neurons



A McCulloch Pitts unit

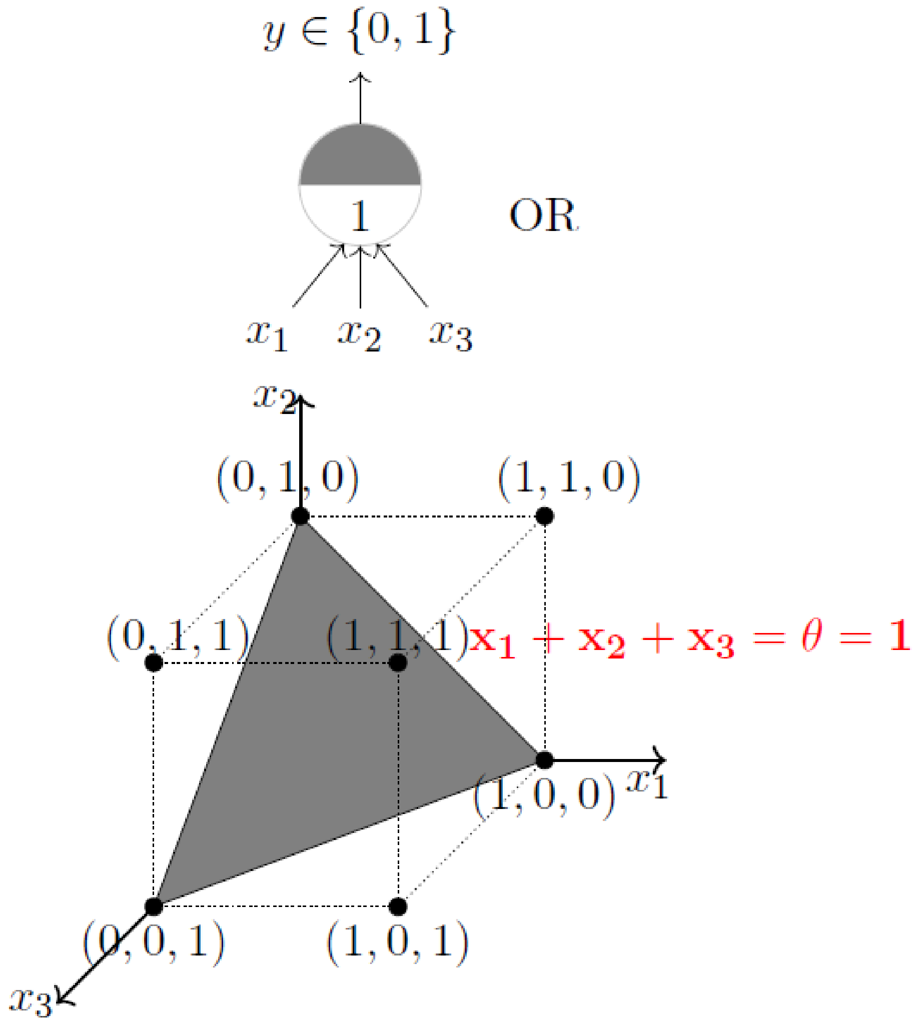


AND function



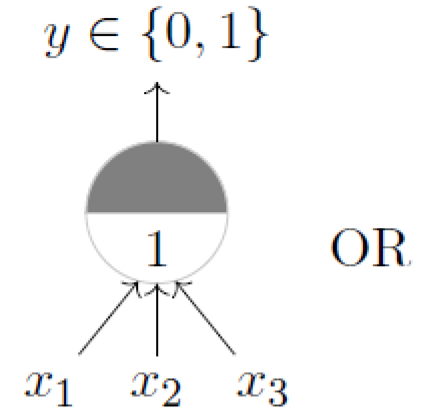
OR function

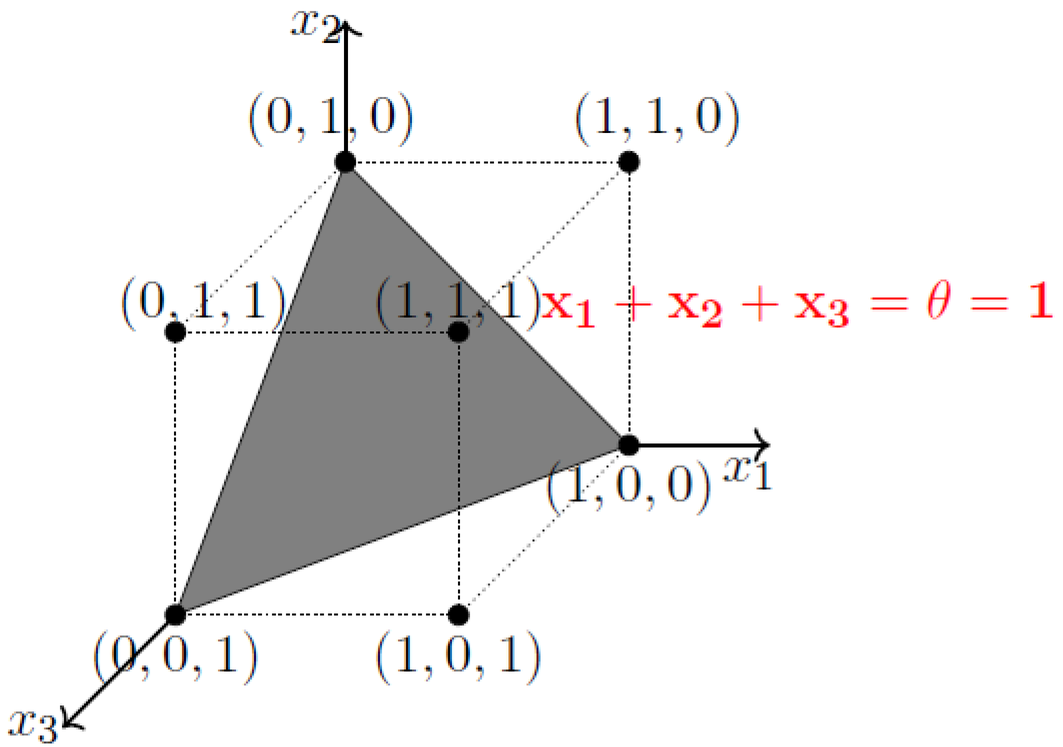
MP Neurons



- What if we have more than 2 inputs?
- Well, instead of a line we will have a plane
- For the OR function, we want a plane such that the point (0,0,0) lies on one side and the remaining 7 points lie on the other side of the plane

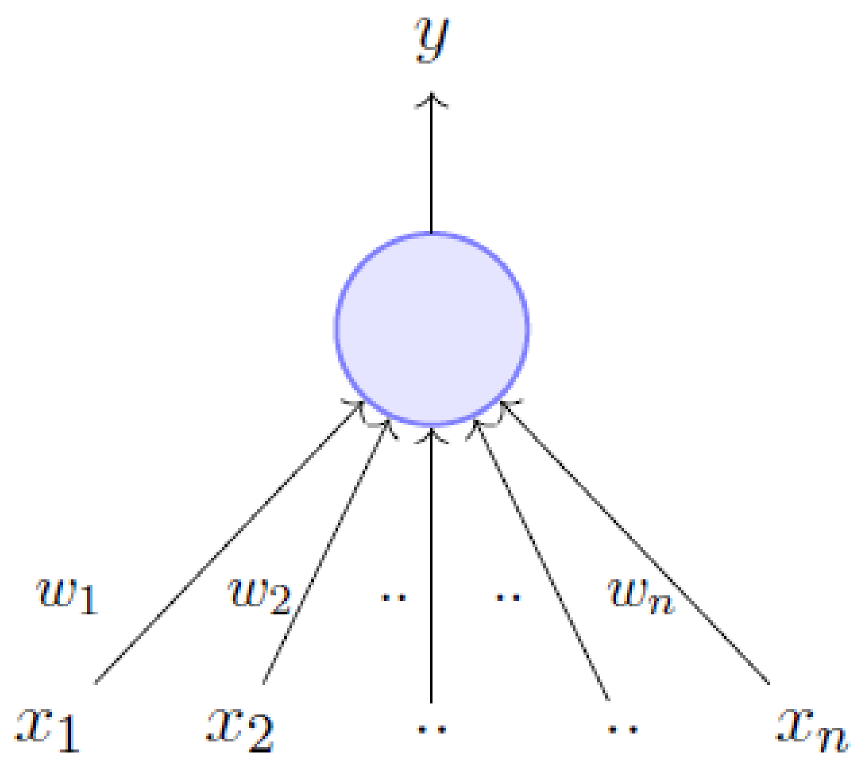
MP Neurons





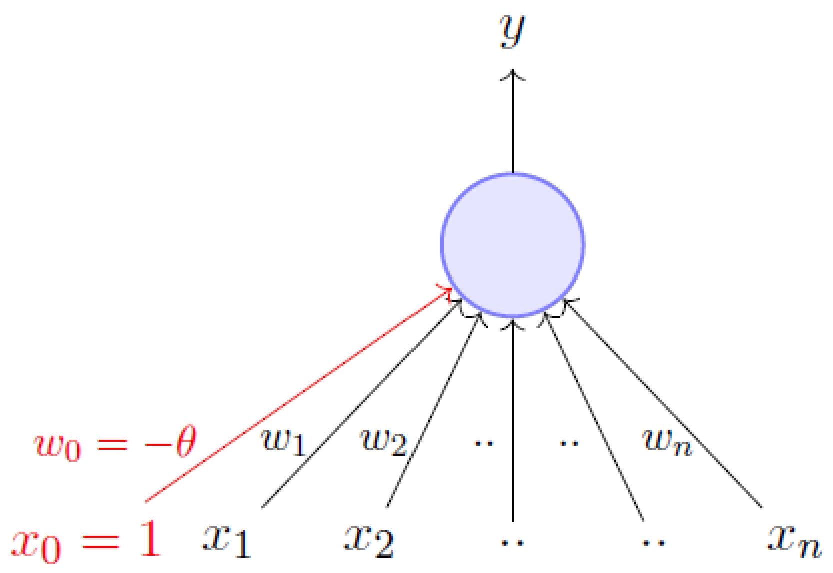
- What if we have more than 2 inputs?
- Well, instead of a line we will have a plane
- For the OR function, we want a plane such that the point (0,0,0) lies on one side and the remaining 7 points lie on the other side of the plane

Perceptron



- Frank Rosenblatt, an American psychologist, proposed the classical perceptron model (1958)
- A more general computational model than McCulloch Pitts neurons
- Main differences: Introduction of numerical weights for inputs and a mechanism for learning these weights.
- Inputs are no longer limited to boolean values
- Refined and carefully analyzed by Minsky and Papert (1969) - their model is referred to as the perceptron model here

Perceptron



$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i \geq \theta$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i < \theta$$

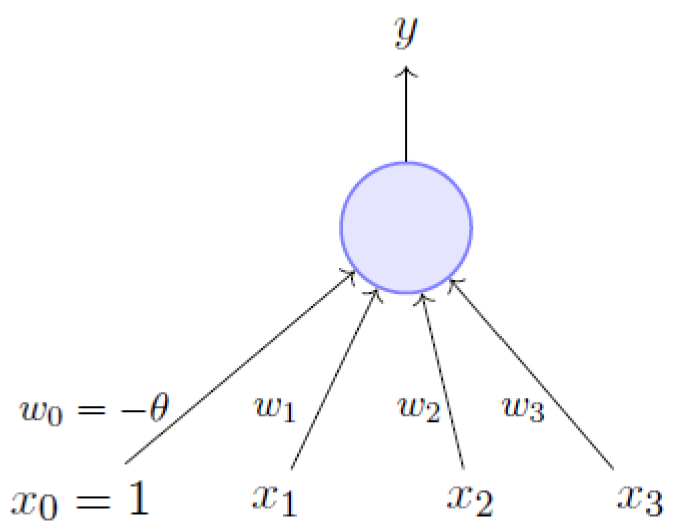
$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i - \theta \geq 0$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i - \theta < 0$$

A more accepted convention,

$$y = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$
$$= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0$$

where; $x_0 = 1$ and $w_0 = -\theta$

Perceptron



x_1 = isActorDamon
x_2 = isGenreThriller
x_3 = isDirectorNolan

1. Consider the task of predicting whether we would like a movie or not.
2. Suppose, we base our decision on 3 inputs (binary, for simplicity)
3. Based on our past viewing experience (data), we may give a high weight to isDirectorNolan as compared to the other inputs.
4. Specifically, even if the actor is not Matt Damon and the genre is not thriller we would still want to cross the threshold theta by assigning a high weight to isDirect orNolan
5. w0 is called the bias
6. A movie buff may have a very low threshold and may watch any movie irrespective of the genre, actor, director [theta = 0]

Key Differences MP Neuron and Perceptron

McCulloch Pitts Neuron (assuming no inhibitory inputs)

$$y = 1 \quad if \sum_{i=0}^{n} x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^{n} x_i < 0$$

Perceptron

$$y = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0$$

- MP Neuron is a simplistic model designed to show how neurons could process binary information using logic gates (AND, OR, NOT).
- Perceptron is a more advanced model that incorporates learning through a training algorithm.
- The MP Neuron does not use weights. All inputs are treated equally, and a threshold determines the output.
- The Perceptron uses weights and biases to prioritize certain inputs and adjust the model based on training data.
- The **Perceptron** is essentially an enhanced and trainable version of the **MP Neuron**, designed for practical machine learning applications.

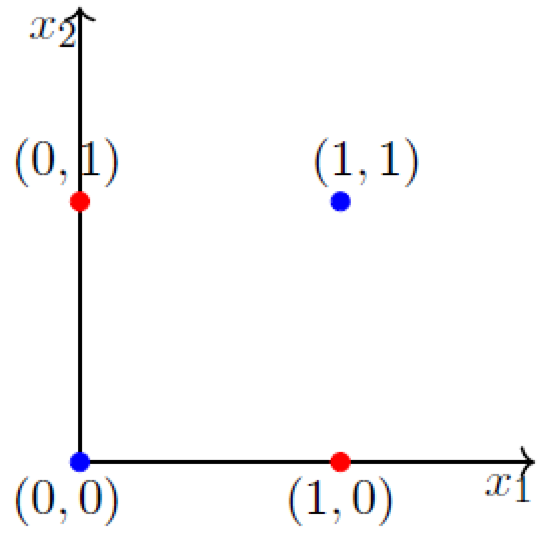
Disadvantages of Perceptrons

So what do we do about functions which are not linearly separable ?
Let us see one such simple boolean 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	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

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

- The fourth condition contradicts conditions 2 and 3.
- Hence we cannot have a solution to this set of inequalities



And indeed you can see that it is impossible to draw a line which separates the red points from the blue points

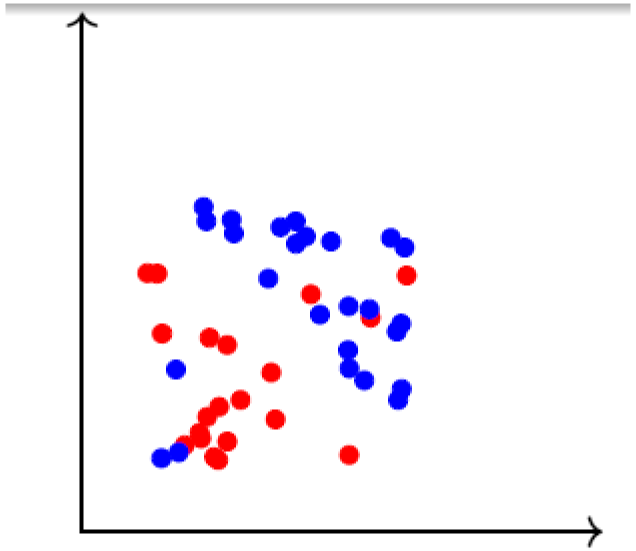
Representation Power of a Network of Perceptrons

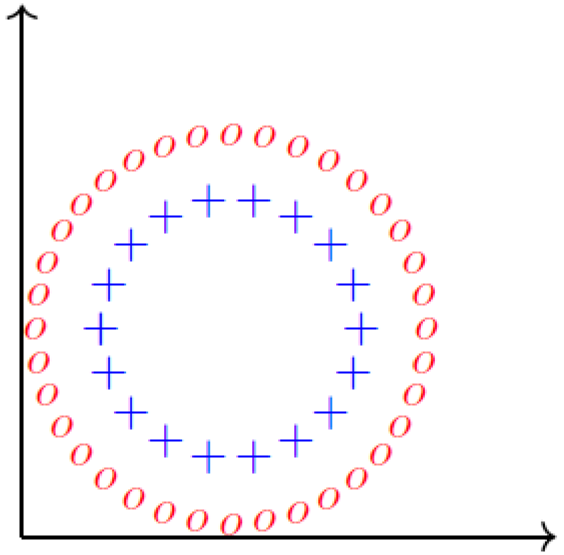
- How many boolean functions can you design from 2 inputs ?
- Let us begin with some easy ones which you already know

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- Of these, how many are linearly separable ? (turns out all except XOR and !XOR - feel free to verify) In general, how many boolean functions can you have for n inputs ?
- How many of these $2^{2^\wedge n}$
- functions are not linearly separable ? For the time being, it suffices to know that at least some of these may not be linearly inseparable

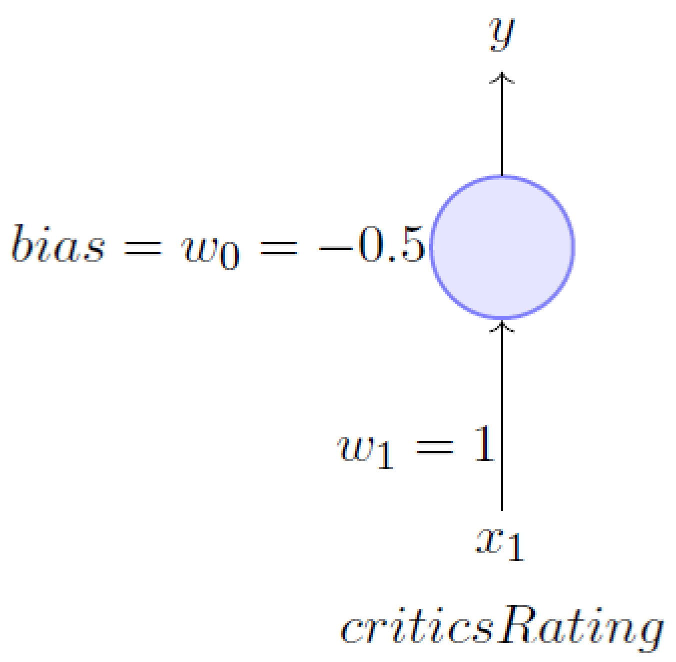
Disadvantages of Perceptrons





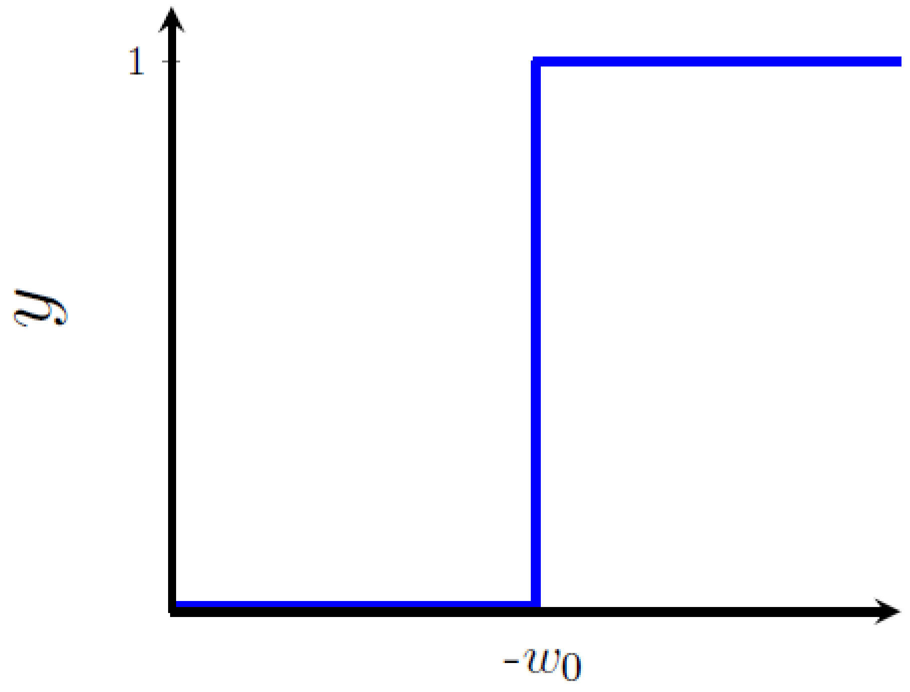
- Most real world data is not linearly separable and will always contain some outliers
- In fact, sometimes there may not be any outliers but still the data may not be linearly separable
- We need computational units (models) which can deal with such data.
- While a single perceptron cannot deal with such data, we will show that a network of perceptron can indeed deal with such data

Representation Power of a Network of Perceptrons



- The thresholding logic used by a perceptron is very harsh !
- we will like or dislike a movie Consider that we base our decision only on one input (x1 = criticsRating which lies between 0 and 1)
- If the threshold is 0.5 (w0 = -0.5) and w1 = 1 then what would be the decision for a movie with criticsRating = 0:51 ? (like)
- What about a movie with criticsRating = 0:49 ? (dislike)
- It seems harsh that we would like a movie with rating 0.51 but not one with a rating of 0.49

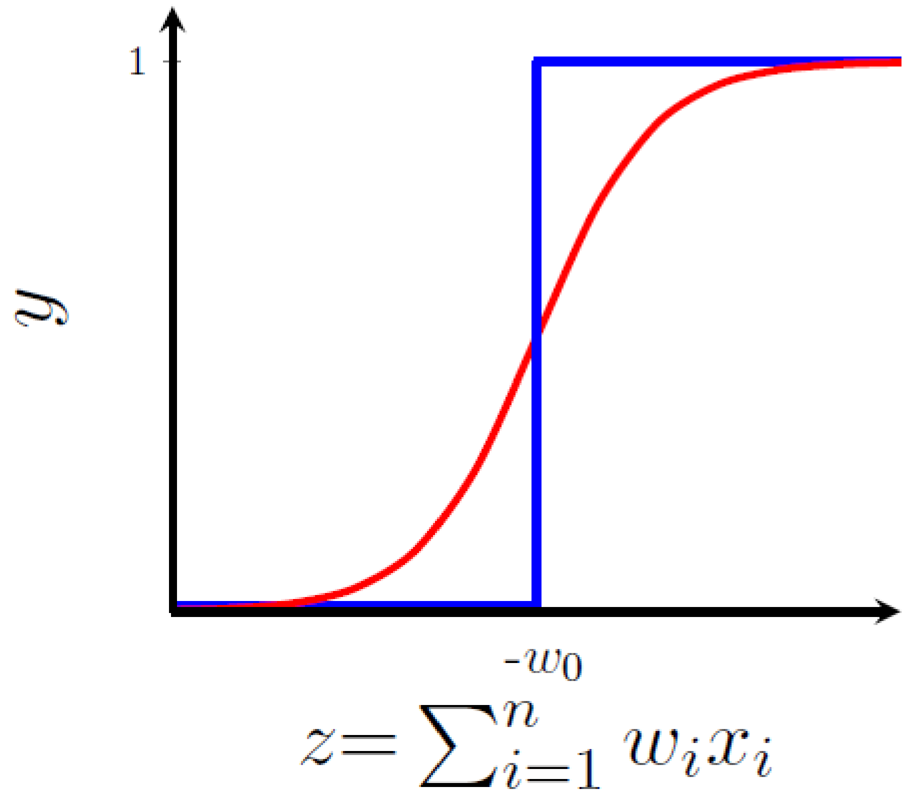
Representation Power of a Network of Perceptrons



$$z = \sum_{i=1}^{n} w_i x_i$$

- This behavior is not a characteristic of the specific problem we chose or the specific weight and threshold that we chose.
- It is a characteristic of the perceptron function itself which behaves like a step function.
- For most real world applications we would expect a smoother decision function which gradually changes from 0 to 1

Representation Power of a Network of Perceptrons

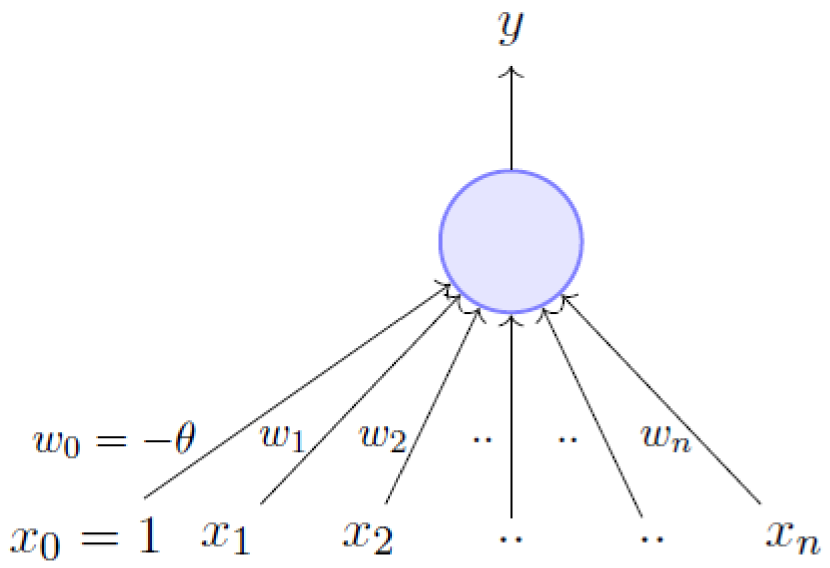


- Introducing sigmoid neurons where the output function is much smoother than the step function
- Here is one form of the sigmoid function called the logistic function
- For most real world applications we would expect a smoother decision function which gradually changes from 0 to 1

$$y = \frac{1}{1 + e^{-(w_0 + \sum_{i=1}^{n} w_i x_i)}}$$

Representation Power of a Network of Perceptrons

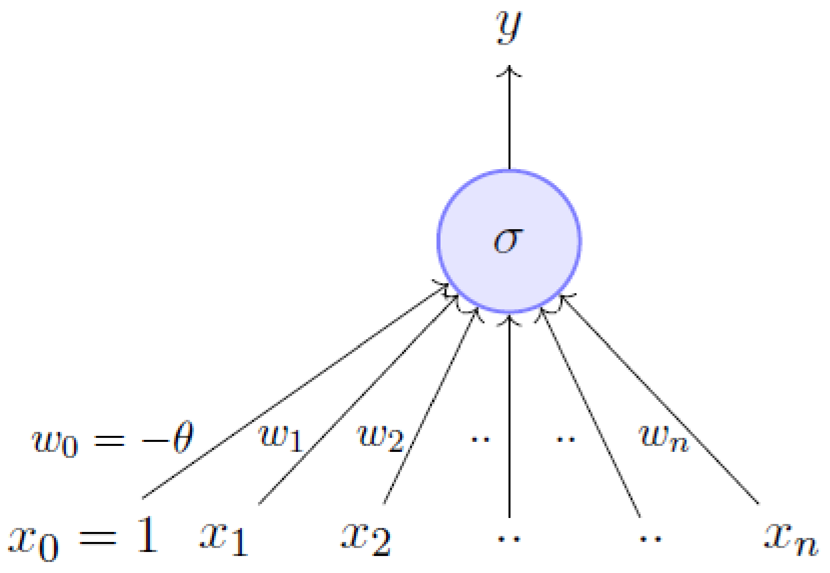
Perceptron



$$y = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$

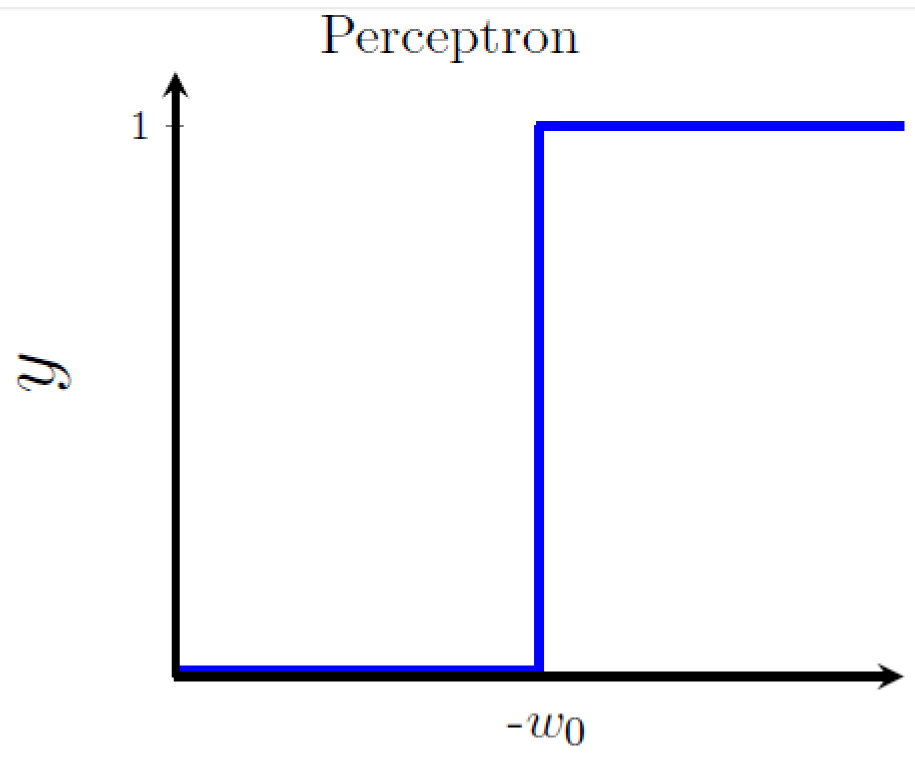
$$= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0$$

Sigmoid (logistic) Neuron



$$y = \frac{1}{1 + e^{-(\sum_{i=0}^{n} w_i x_i)}}$$

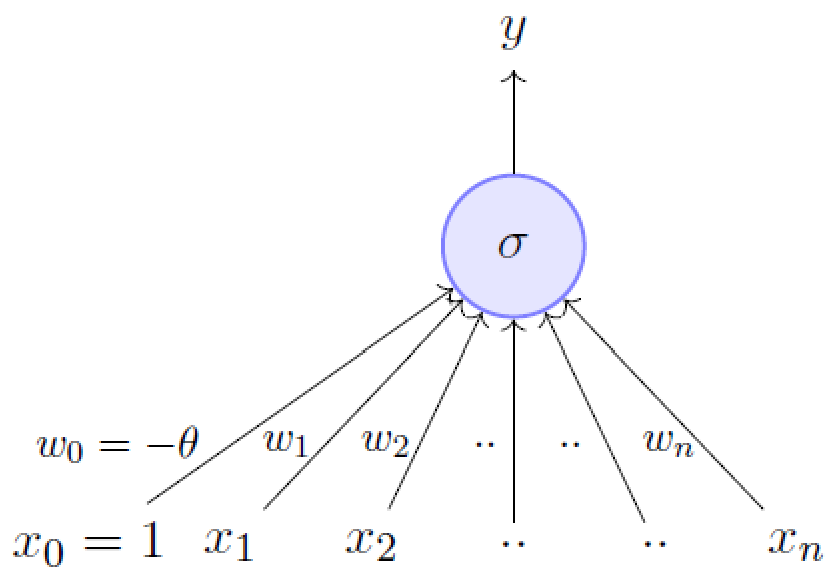
Representation Power of a Network of Perceptrons



$$z = \sum_{i=1}^{n} w_i x_i$$

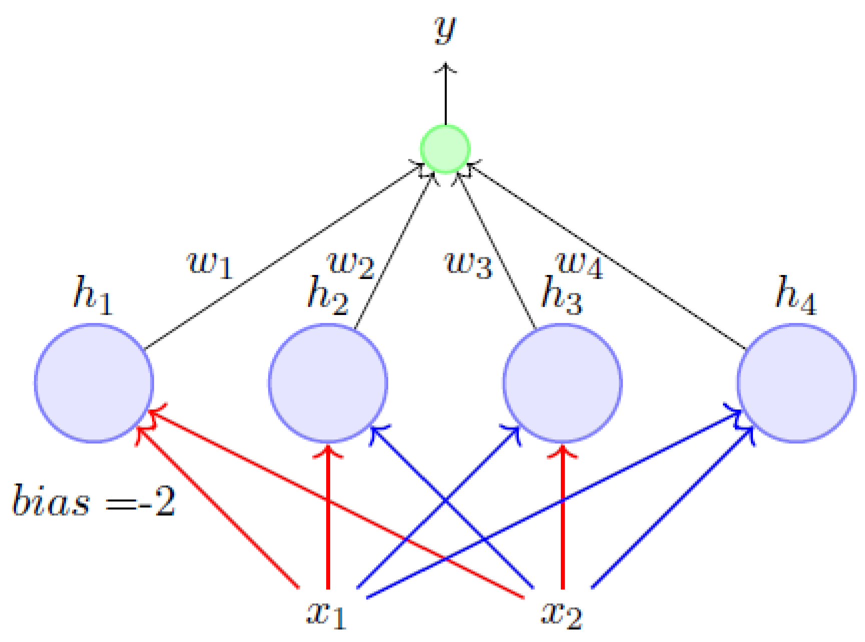
Not smooth, not continuous (at w0), not differentiable

Sigmoid (logistic) Neuron



$$y = \frac{1}{1 + e^{-(\sum_{i=0}^{n} w_i x_i)}}$$

Terminology

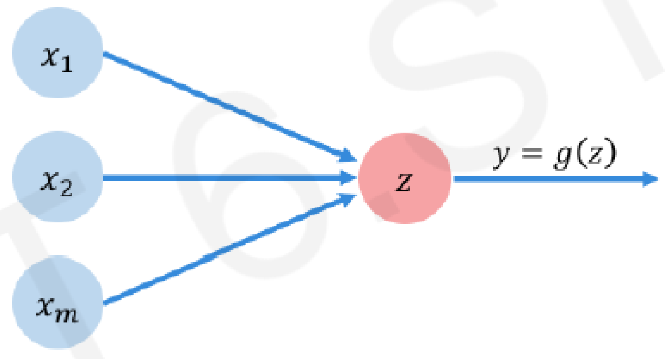


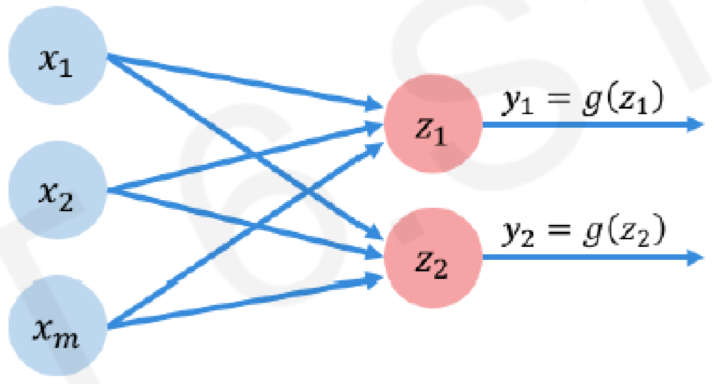
red edge indicates w = -1
blue edge indicates $w = +1$

- This network contains 3 layers
- The layer containing the inputs (x1, x2) is called the input layer
- The middle layer containing the 4 perceptrons is called the hidden layer.
- The final layer containing one output neuron is called the output layer.
- The outputs of the 4 perceptrons in the hidden layer are denoted by h1, h2, h3, h4.
- The red and blue edges are called layer 1 weights.
- W1,w2,w3,w4 are called layer 2 weights.

Terminology

- A Multilayer Perceptron (MLP) is one of the simplest and most common neural network architectures used in machine learning.
- It is a **feedforward artificial neural network** consisting of multiple layers of interconnected neurons, including an **input layer**, one or more **hidden layers**, and an **output layer**.
- In the context of Deep Learning, a **Perceptron** is usually referred to as a neuron, and a Multi-Layer Perceptron structure is referred to as a Neural Network.
- MLPs are capable of learning complex and non-linear relationships in data especially when they have multiple hidden layers and non-linear activation functions.





$$z = w_0 + \sum_{j=1}^{m} x_j \, w_j$$

$$z_i = w_{0,i} + \sum_{j=1}^{m} x_j \, w_{j,i}$$

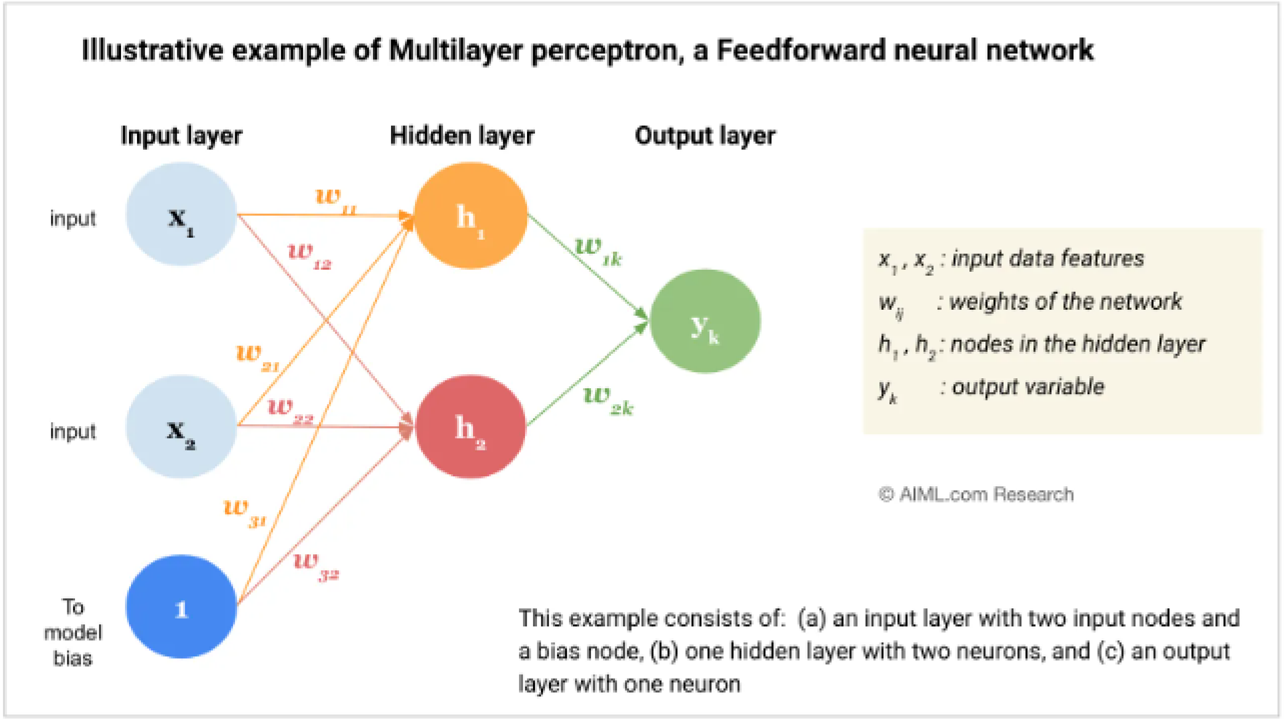
The key characteristics and training process of an MLP model:

A Multi-Layer Perceptron learns **to map inputs to outputs** through a process of **forward propagation and backpropagation**, **adjusting its weights** and biases based on the **error** between its **predictions** and the **actual** data.

1. **Input layer:** This receives the input data. Each neuron in this layer represents a feature of the input data.

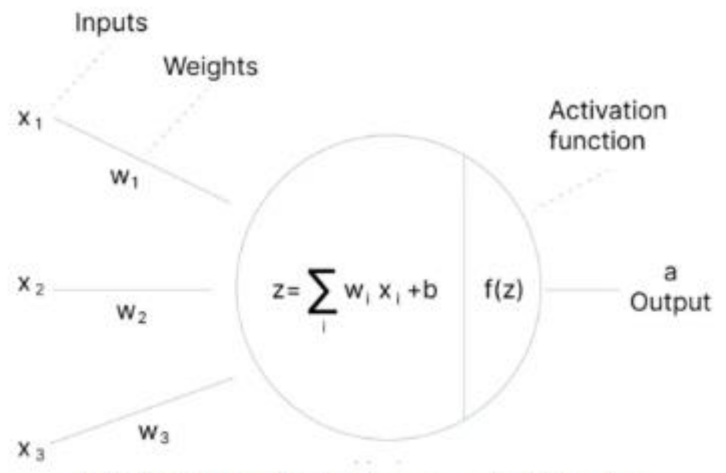
2. **Weighted Connections and Biases**
   - Connections between neurons have **associated weights**, which are learned during the training process.
   - These weights determine the **strength of the connections** and play a crucial role in the network's ability **to capture patterns** in the data.
   - In addition, each neuron, in the hidden and output layers has an associated **bias** term, which allows for fine-tuning and shifting the activation function's threshold.
   - These weights and biases are parameters that the neural network **learns during training.**

The key characteristics and training process of an MLP model:



x_1, x_2	: *input data features*
w_{ij}	: *weights of the network*
h_1, h_2	: *nodes in the hidden layer*
y_k	: *output variable*

This example consists of: (a) an input layer with two input nodes and a bias node, (b) one hidden layer with two neurons, and (c) an output layer with one neuron



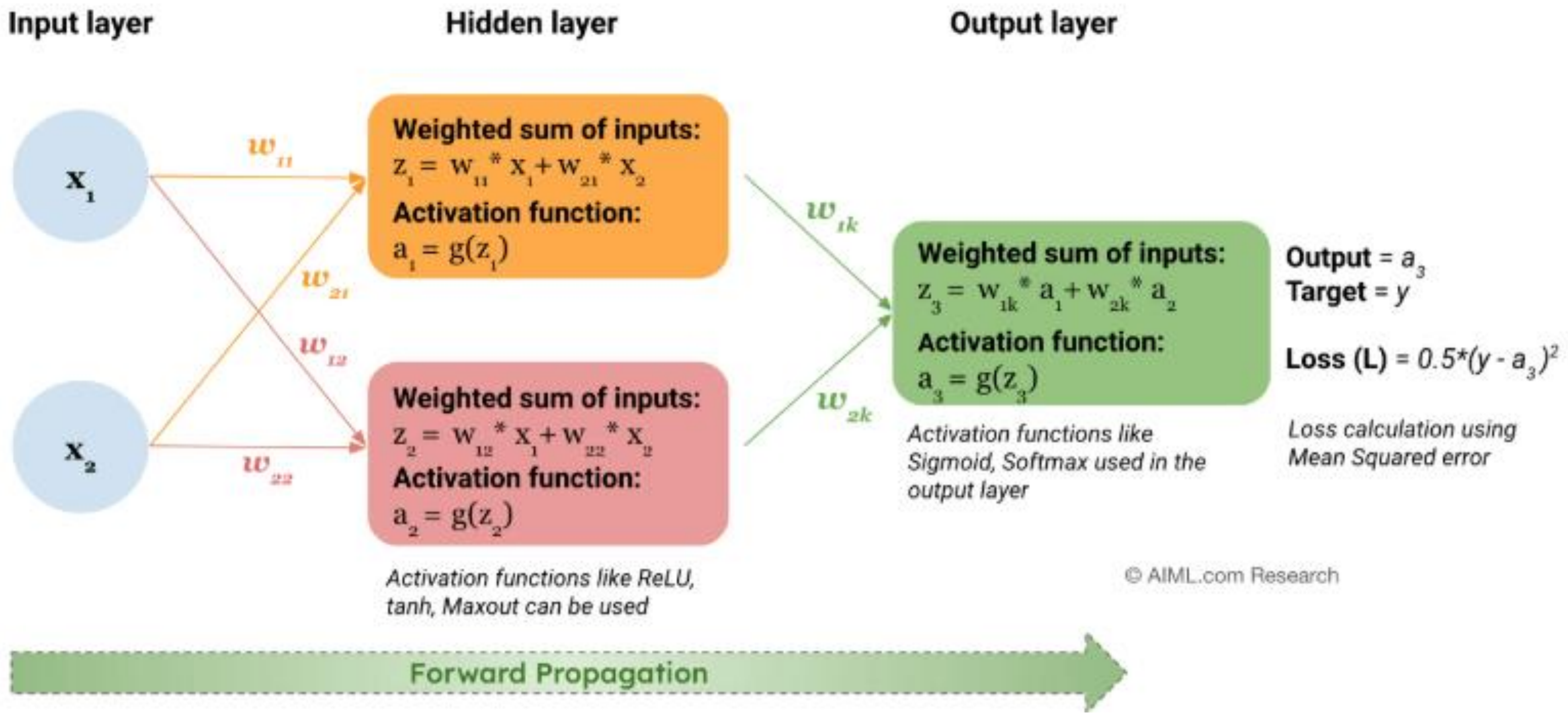
Depicting input and output to a neuron in hidden layers

The key characteristics and training process of an MLP model:

3. **Activation Functions**
   The activation function is crucial as it introduces non-linearity into the model, allowing it to learn more complex patterns. Common activation functions include sigmoid, tanh, and ReLU (Rectified Linear Unit).

4. **Forward Propagation**
   - The process from the input layer through the hidden layers to the output layer is called **forward propagation.**
   - In each layer, the aforementioned steps (**weighted sum**, **bias addition**, **activation function**) are applied to compute the layer's output.
   - In an MLP, information flows in one direction, from the input layer through the hidden layers to the output layer.
   - There are no feedback loops or recurrent connections, hence the name feedforward architecture.

The key characteristics and training process of an MLP model:

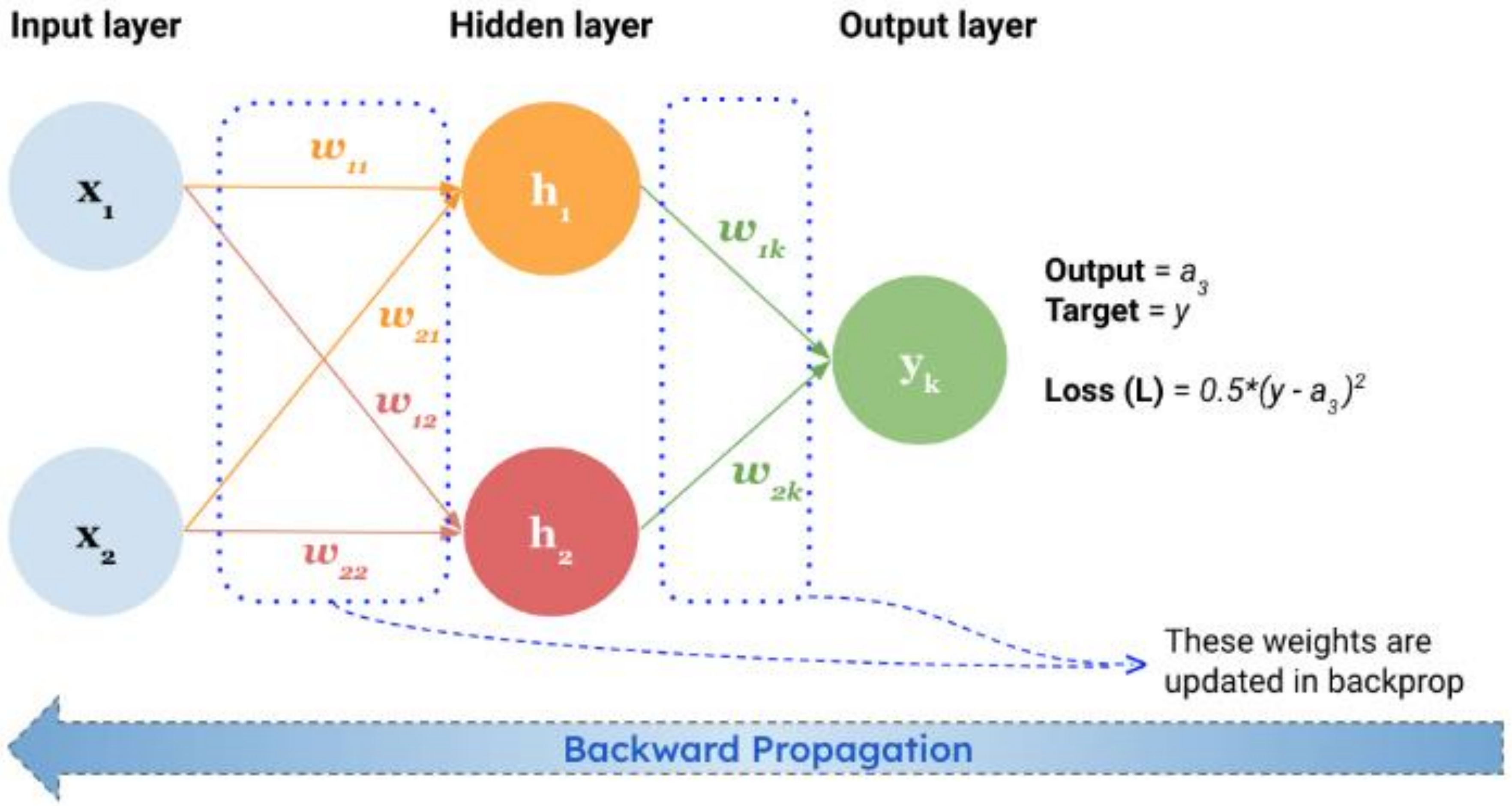


The key characteristics and training process of an MLP model:

3. **Output layer**
   The final layer is the output layer. In a classification task, this layer often uses a **softmax** function if the task is **multi-class classification**, or a sigmoid function **for binary classification**.

4. **Backpropagation and Learning**
   - Once a forward pass through the network is completed, the output is compared to the true value to calculate the error.
   - The error is then propagated back through the network (backpropagation), adjusting the weights and biases to minimize the error. Backpropagation is typically done using optimization algorithms, such as stochastic gradient descent (SGD) and its variants.

The key characteristics and training process of an MLP model:



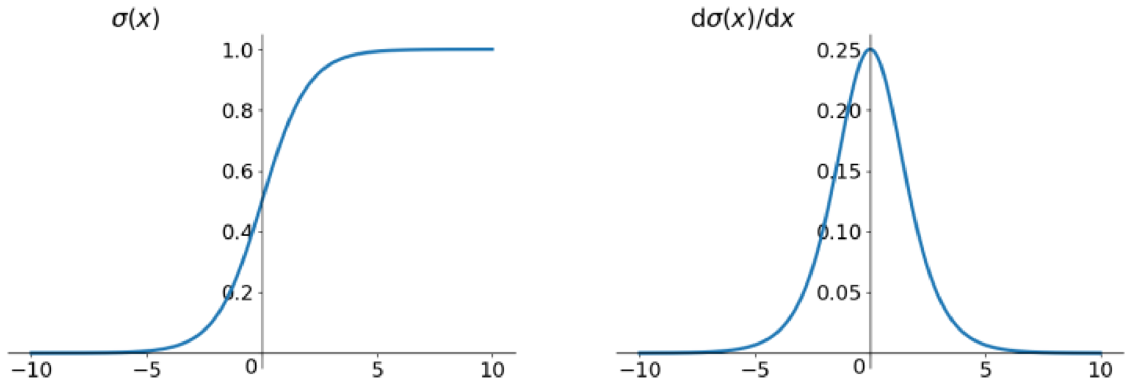
The key characteristics and training process of an MLP model:

5. **Iteration and Convergence:** The process of forward propagation, error calculation, backpropagation, and parameter update is repeated for many iterations over the training data. Gradually, the network learns to reduce the error, and the weights and biases converge to values that make the network capable of making accurate predictions or approximations.

Activation Functions

1. **Sigmoid Activation Function** is characterized by 'S' shape

- $\sigma(x) = sigmoid(x) = \frac{1}{1+e^{-x}}$
- $\sigma'(x) = \frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$



Advantages

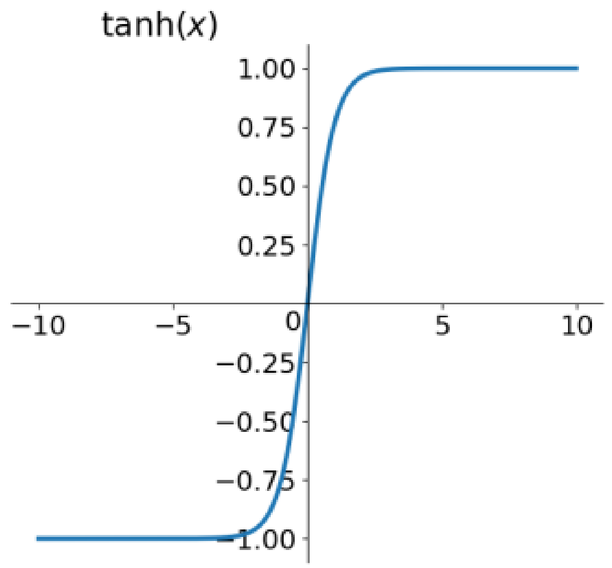
- It allows neural networks to model complex patterns that linear equations cannot.
- Converts inputs into probabilities.
- The output ranges between 0 and 1, hence useful for binary classification.

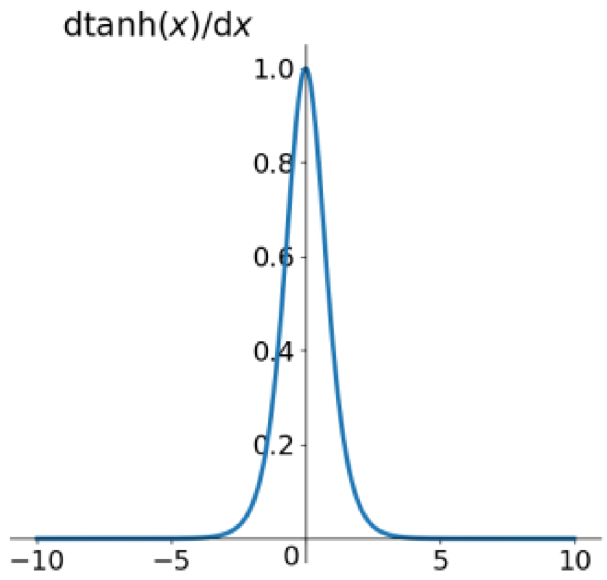
Disadvantages

- Prone to vanishing gradient problems for large or small inputs. (the derivatives are less than 1 for most input values. Multiplying many such small derivatives causes the gradient to shrink exponentially as it moves backward through layers.)
- Outputs are not zero-centered. (The optimization algorithm may take longer to converge because the updates are not symmetrically balanced, which can lead to oscillations or inefficiency.)

tanh function (hyperbolic tangent function)

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$





Range: (−1,1)

- tanh function is used for the hidden layer.
- sigmod function is used for the output layer.

Advantages

- Zero-centered, making optimization easier compared to sigmoid.

Disadvantages

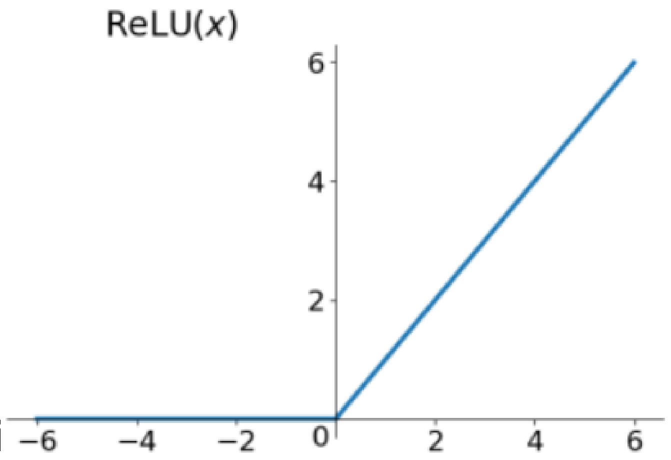
- Prone to vanishing gradient problems for large or small inputs.

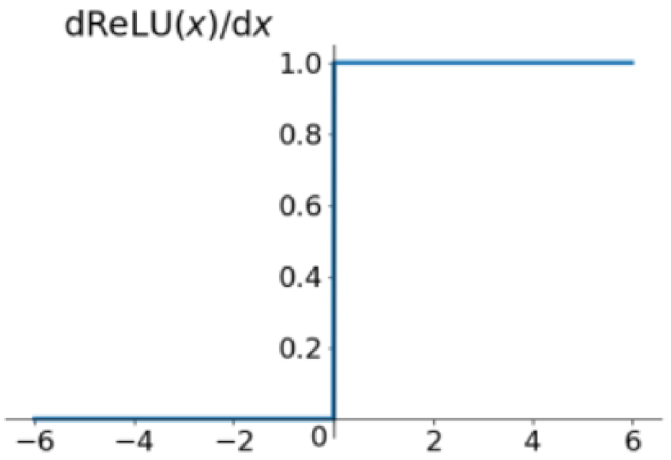
ReLU function (Rectified Linear Unit)

A(x)=max(0,x)

Range: [0,∞)

- The most commonly used activation functi
  in hidden layers.





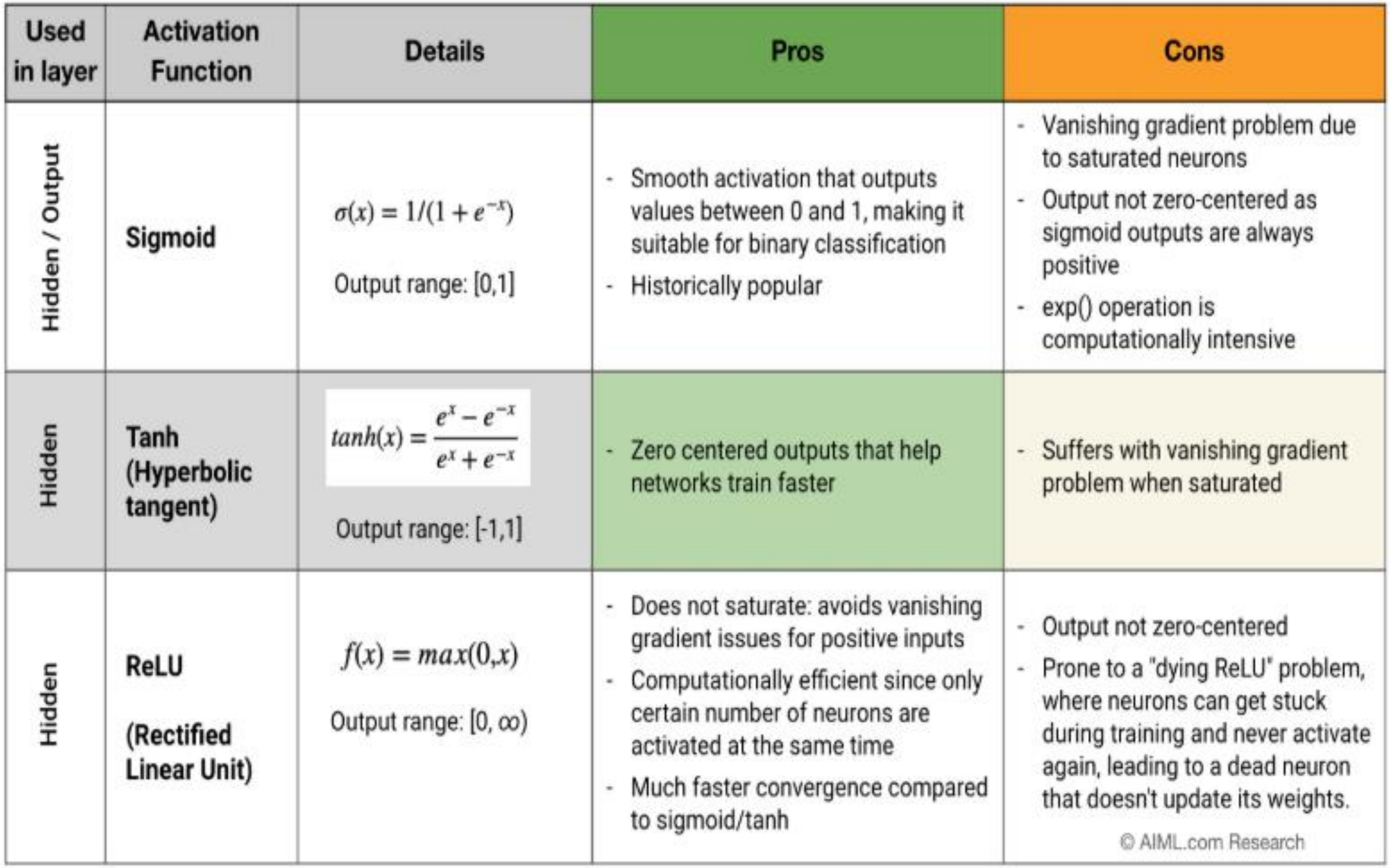
Advantages:

- Computationally efficient.
- Mitigates vanishing gradient issues.

Disadvantages

- Can result in "dead neurons" (neurons that output 0 for all inputs).

Sigmoid, Tanh, and ReLU

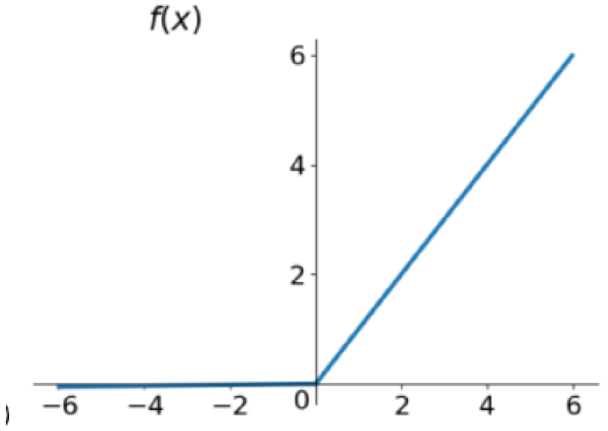


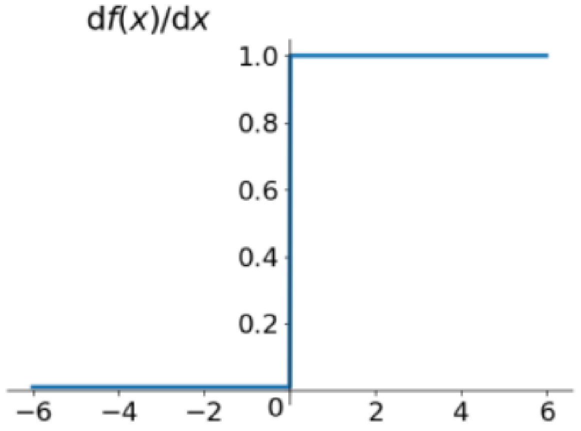
Used in layer	Activation Function	Details	Pros	Cons
Hidden / Output	**Sigmoid**	$\sigma(x) = 1/(1+e^{-x})$ Output range: [0,1]	- Smooth activation that outputs values between 0 and 1, making it suitable for binary classification - Historically popular	- Vanishing gradient problem due to saturated neurons - Output not zero-centered as sigmoid outputs are always positive - exp() operation is computationally intensive
Hidden	**Tanh (Hyperbolic tangent)**	$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ Output range: [-1,1]	- Zero centered outputs that help networks train faster	- Suffers with vanishing gradient problem when saturated
Hidden	**ReLU** **(Rectified Linear Unit)**	$f(x) = max(0,x)$ Output range: [0, ∞)	- Does not saturate: avoids vanishing gradient issues for positive inputs - Computationally efficient since only certain number of neurons are activated at the same time - Much faster convergence compared to sigmoid/tanh	- Output not zero-centered - Prone to a "dying ReLU" problem, where neurons can get stuck during training and never activate again, leading to a dead neuron that doesn't update its weights.

Leaky ReLU function

$$f(x) = \max(0.01x, x)$$

Range: (−∞,∞)





Advantages:

- Allows a small gradient for negative inputs, reducing the risk of dead neurons.
- Solves the dying neuron problem in ReLU.

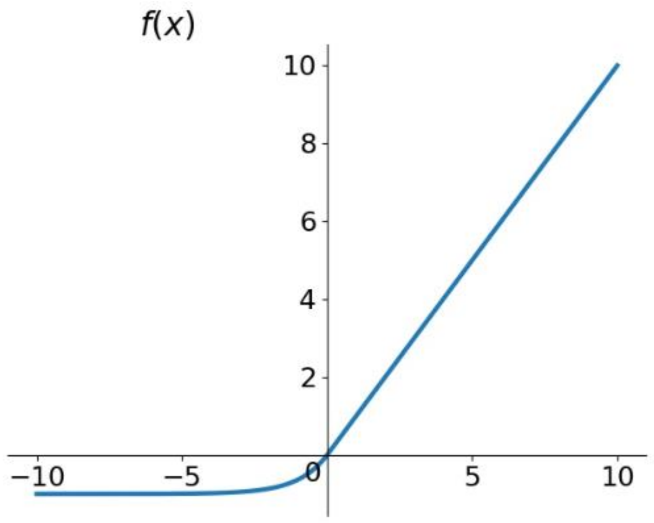
Disadvantages

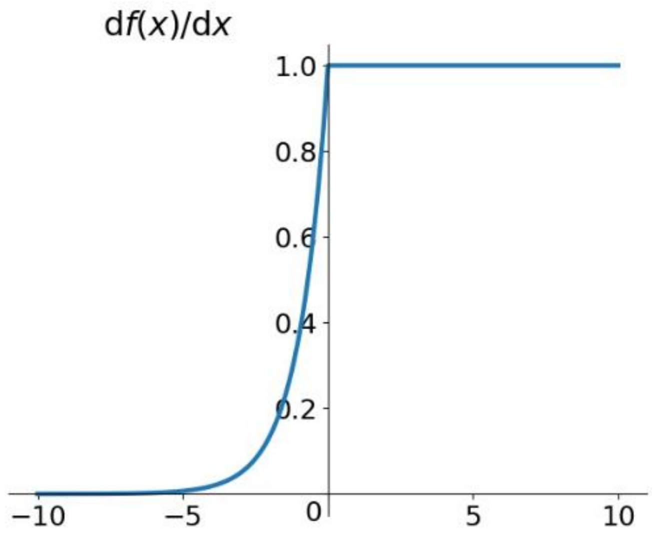
- Introduces a small computational overhead.

ELU (Exponential Linear Units) function

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases}$$

Range: (−alpha,∞)





Advantages:

- Allows a small gradient for negative inputs, reducing the risk of dead neurons.
- Solves the dying neuron problem in ReLU.
- Zero centereed

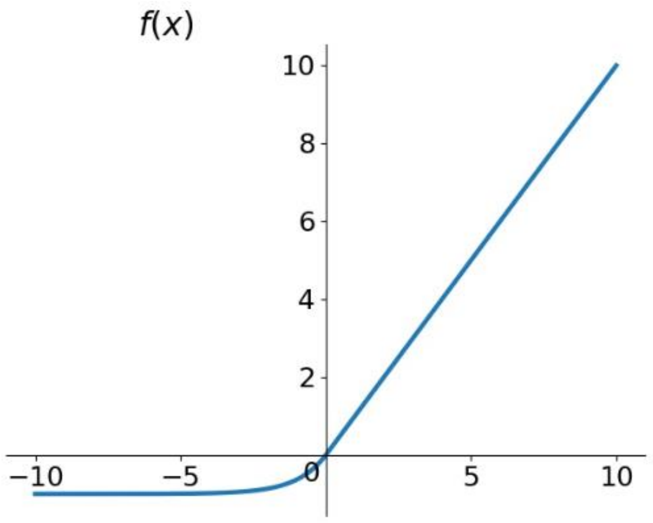
Disadvantages

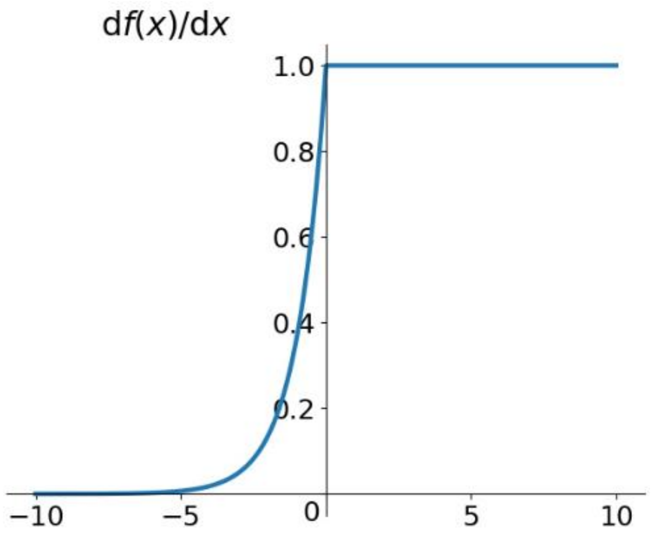
- computational overhead.

Parametric ReLU (PReLU)

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

Range: (−∞,∞)
if a_i=0, f becomes ReLU
if a_i>0, f becomes leaky ReLU
if a_i is a learnable parameter, f becomes PR





Advantages:
- Similar to Leaky ReLU but with learnable parameters.
- Can adapt to different datasets during training.

Disadvantages
- Adds extra parameters to the model.

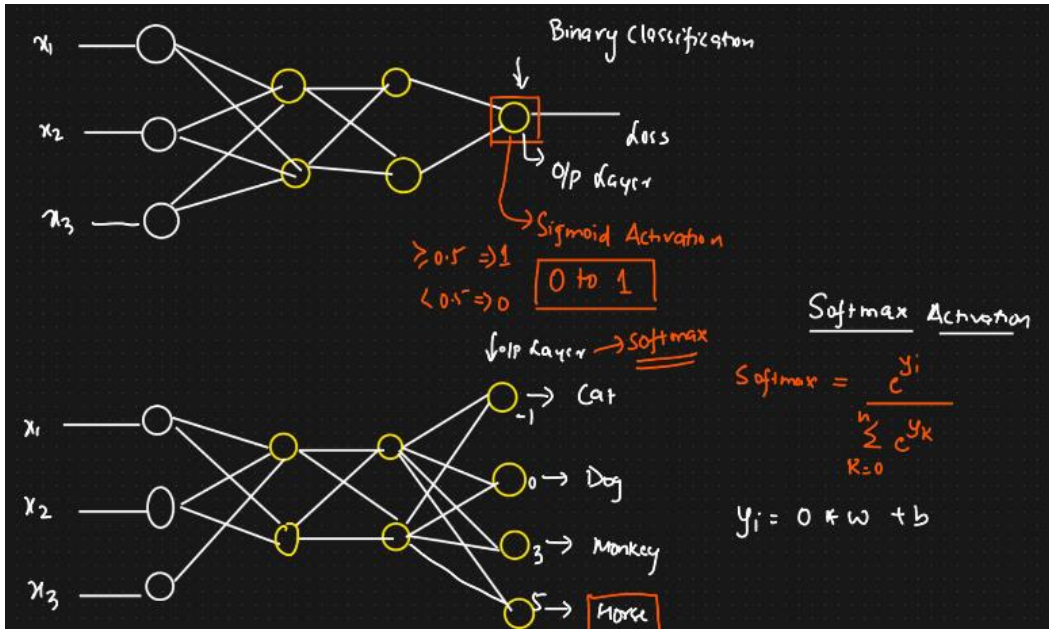
Softmax

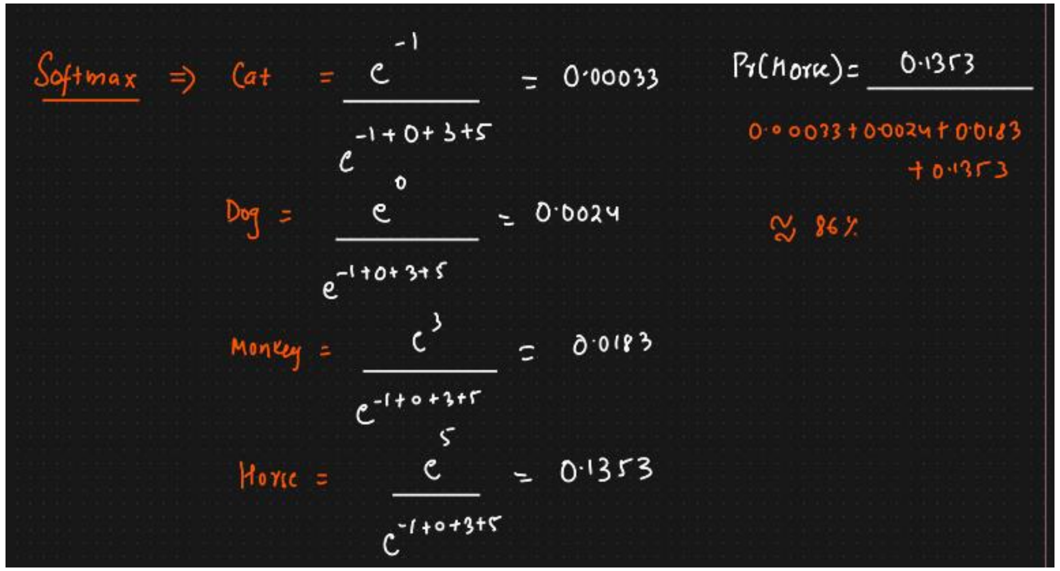
$$S(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, j = 1, 2, \ldots, K$$

Range: (0,1)

Advantages:

- Allows a small gradient for negative inputs, reducing the risk of dead neurons.
- Solves the dying neuron problem in ReLU.
- Zero centereed





Softmax ⇒ Cat $= \frac{e^{-1}}{e^{-1+0+3+5}} = 0.00033$

Dog $= \frac{e^{0}}{e^{-1+0+3+5}} = 0.0024$

Monkey $= \frac{e^{3}}{e^{-1+0+3+5}} = 0.0183$

Horse $= \frac{e^{5}}{e^{-1+0+3+5}} = 0.1353$

Pr(Horse) $= \frac{0.1353}{0.00033 + 0.0024 + 0.0183 + 0.1353}$

$\approx 86\%$

Used in layer	Activation Function	Details	Pros	Cons
Hidden	**Leaky ReLU**	$f(x) = max(0.01x, x)$ Output range: $(-\infty, \infty)$	- All benefits of ReLU - Addresses the "dying ReLU" problem by allowing a small gradient for negative inputs. Helps with training deeper networks	- Not as standardized as ReLU, and the slope of the leaky part is typically a hyperparameter (**α**) that needs tuning. This is also referred to as Parametric ReLU $f(x) = max(\alpha x, x)$
Hidden	**ELU (Exponential Linear Unit)**	where **α** > 0: $f(x) = \begin{cases} x & for\ x \geqslant 0 \\ \alpha(e^x - 1) & for\ x < 0 \end{cases}$ **α** is commonly chosen as 1 Output range: $(-\alpha, \infty)$	- All benefits of ReLU - Zero centered outputs that help networks train faster - ELUs saturate to a negative value when the argument gets smaller becoming more robust to noise	- Computationally more expensive due to exponential operation
Output	**Softmax**	For a given class *i*, probability $P(y_i)$ is: $P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$ where, z_i is the raw score (logit) for class *i*, and *N* is the no. of classes Output range: (0,1)	- Used in the output layer of multi-class classification problems, converting model outputs into probability distributions.	- Not suitable for multi-label classification, as it enforces that only one class can be predicted. © AIML.com Research