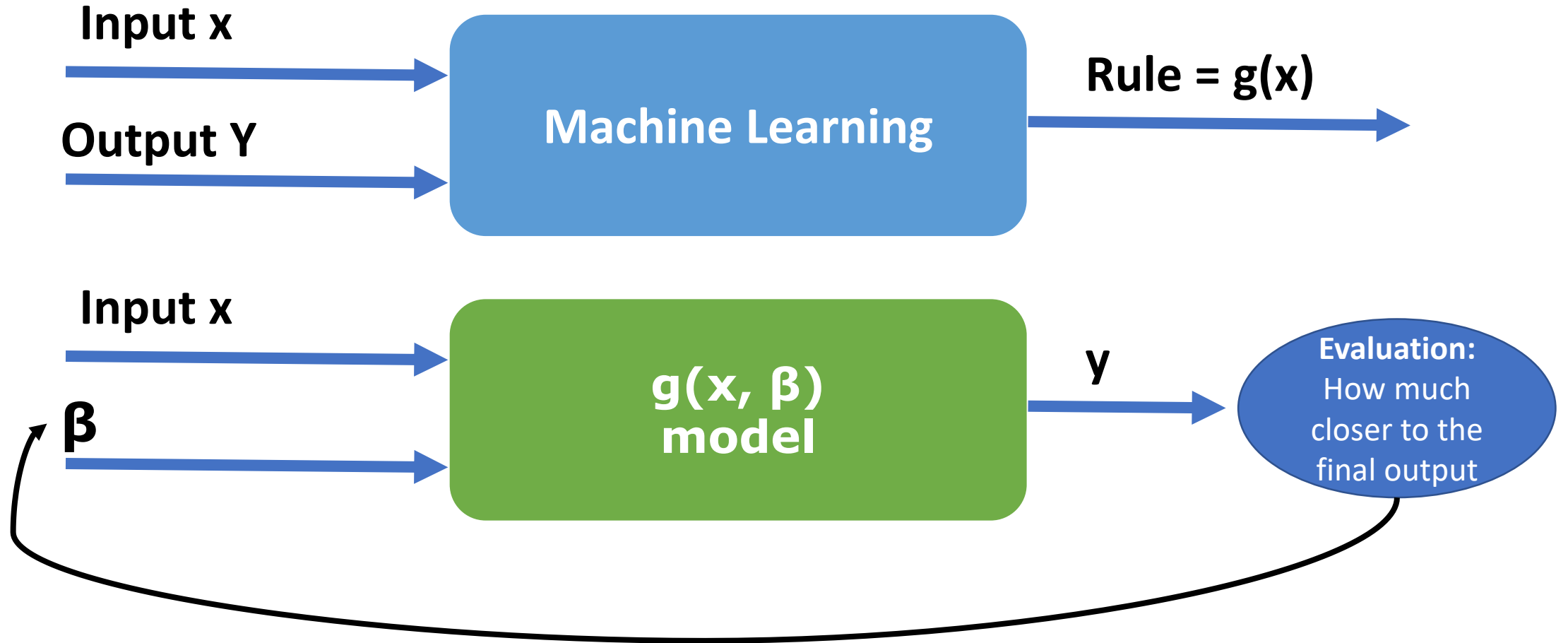


Machine Learning Concept



Supervised Learning, Perceptron

Dr. Mohamed Mostafa

AI 511 Foundations of Artificial Intelligence

Lecture 9



Outline

- Machine Learning Concept
- What is a Perceptron?
 - Types of Perceptron
 - Basic Components of Perceptron
 - Activation function
 - Linear vs non-linear relationship
 - Applications of Single layer perceptron
 - Multilayer perceptions
 - Examples

Resources for this lecture

- Data Mining And Knowledge Discovery Handboook,

Published at

<https://www.researchgate.net/publication/225237661> Decision Tree
s.

- Michael Nielsen, Neural Networks and Deep Learning,

<http://neuralnetworksanddeeplearning.com/chap2.html>

.. AND ..

- Introduction to Artificial Intelligence Course, Dr. Amr S. Ghoneim

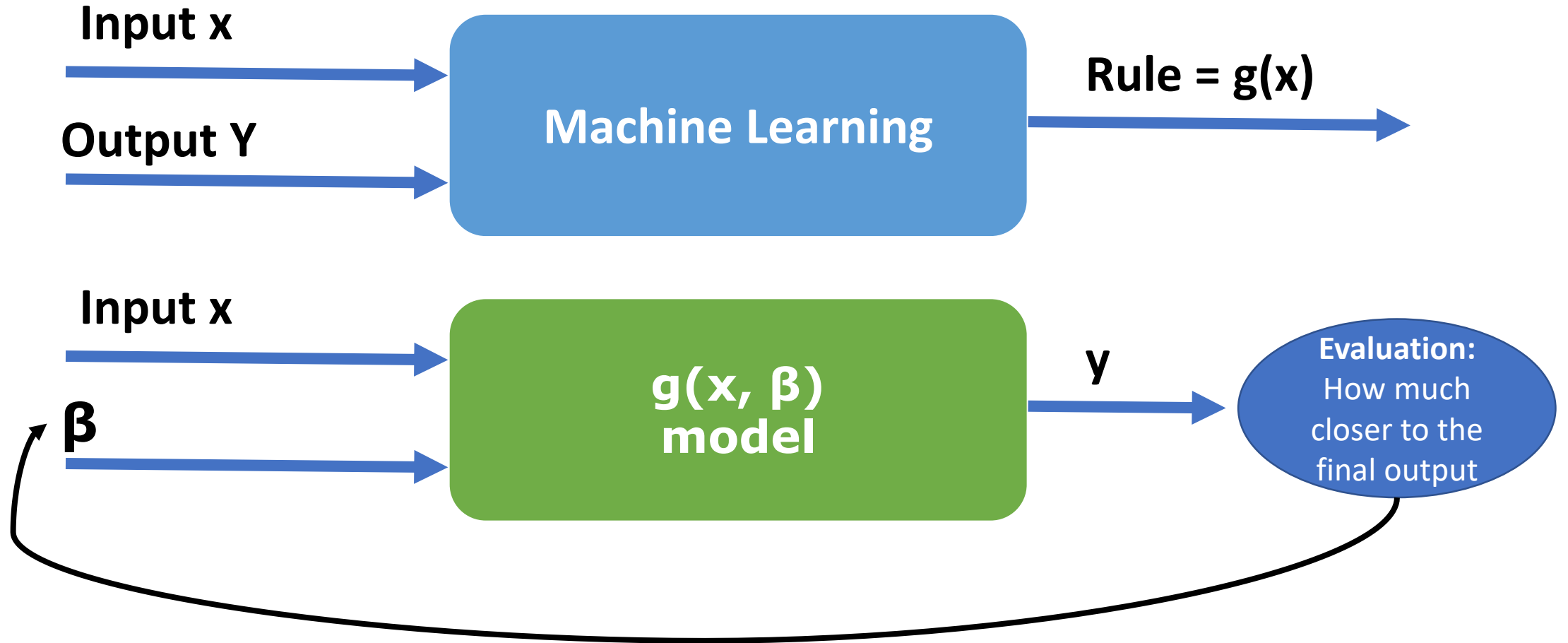
ECE595/STAT598:MachineLearningI Lecture18Multi-LayerPerceptron, Stanley Chan School of Electrical and Computer Engineering Purdue University

Machine Learning Concept

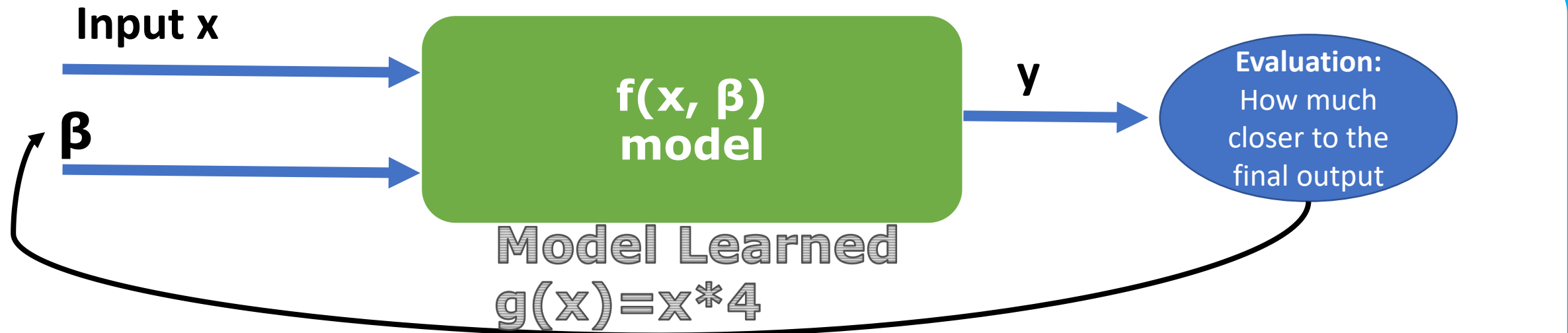


Input $x = 2$;
 Rule = $x * x$;
 $y = f(x) = 2 * 2 = 4$

Machine Learning Concept



Machine Learning Concept



Input $x = 2$;
Output $y = 8$
 $y = 2 * \beta = 8$

$y = 2 * 1 = 2$
 $\text{eval} = 8 - 2 = 6$ (difference is 6)

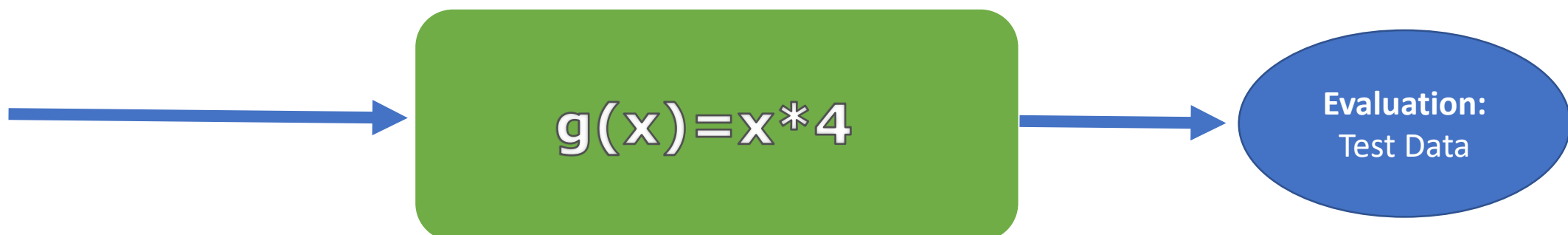
$y = 2 * 2 = 4$
 $\text{eval} = 8 - 4 = 4$ (difference is 4)

$y = 2 * 3 = 6$
 $\text{eval} = 8 - 6 = 2$ (difference is 2)

$y = 2 * 4 = 8$
 $\text{eval} = 8 - 8 = 0$ (difference is 0)

Input Testing Data

**x1,
x2,
x3,
x4,**


$$g(x) = x * 4$$

Output Testing Data

**y1,
y2,
y3,
y4,**

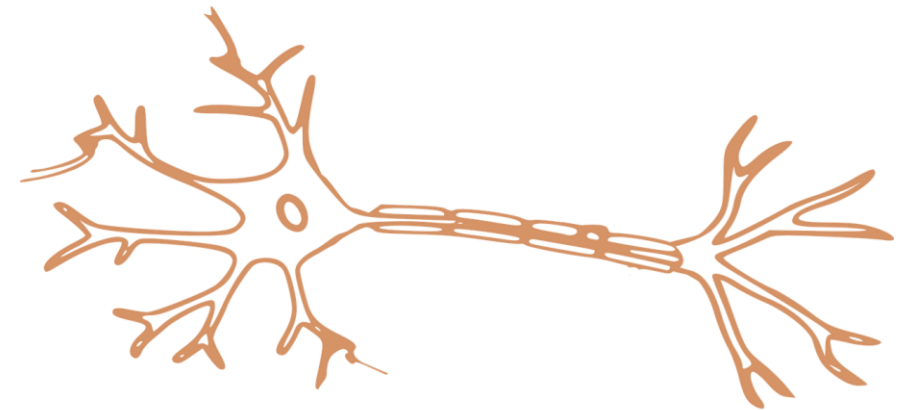
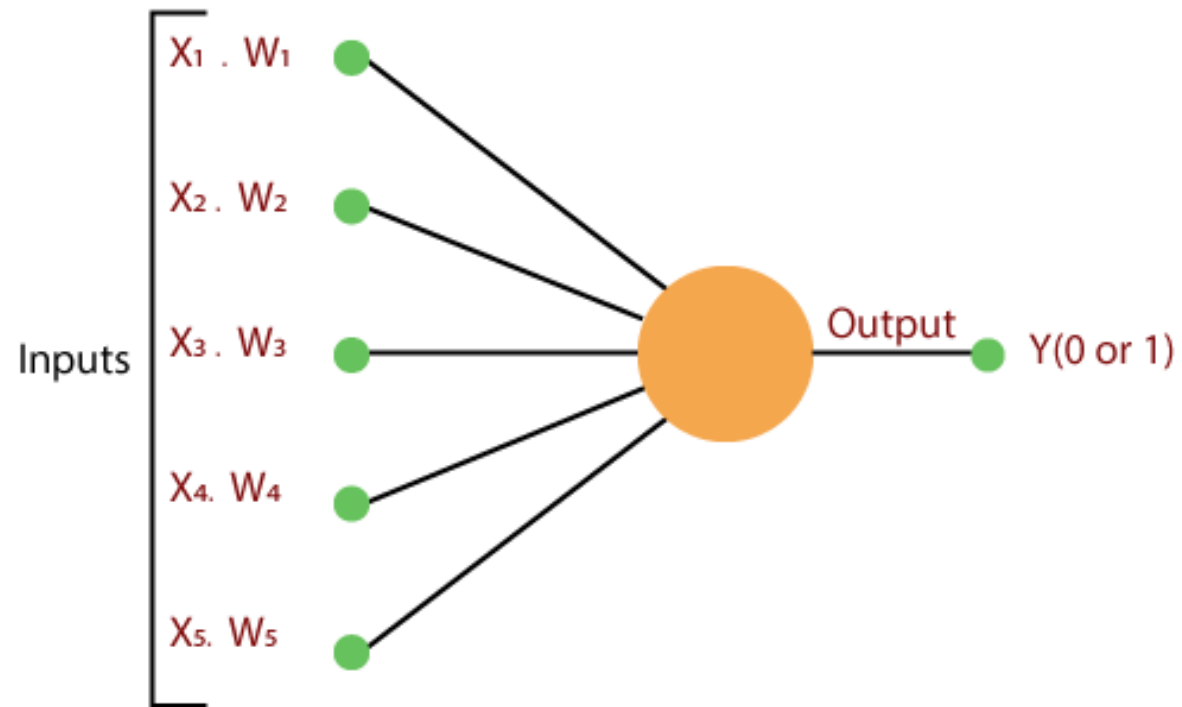
**Evaluation:
Test Data**

What is a Perceptron?

- A **Perceptron** is an **Artificial Neuron**
- It is the simplest possible **Neural Network**
- **Historically:**
 - Frank Rosenblatt (1928 – 1971) was a prominent American psychologist known for his contributions to the field of Artificial Intelligence.
 - In 1957, he initiated a significant development by creating the Perceptron program, which ran on an IBM 704 computer at Cornell Aeronautical Laboratory.
 - Researchers had observed that neurons in the brain receive input from sensory organs through electrical signals.
 - Subsequently, neurons utilize electrical signals to store information and make decisions based on past input.
 - Rosenblatt conceived the idea that Perceptrons could emulate brain functions, possessing the capacity to learn from input data and make decisions autonomously.

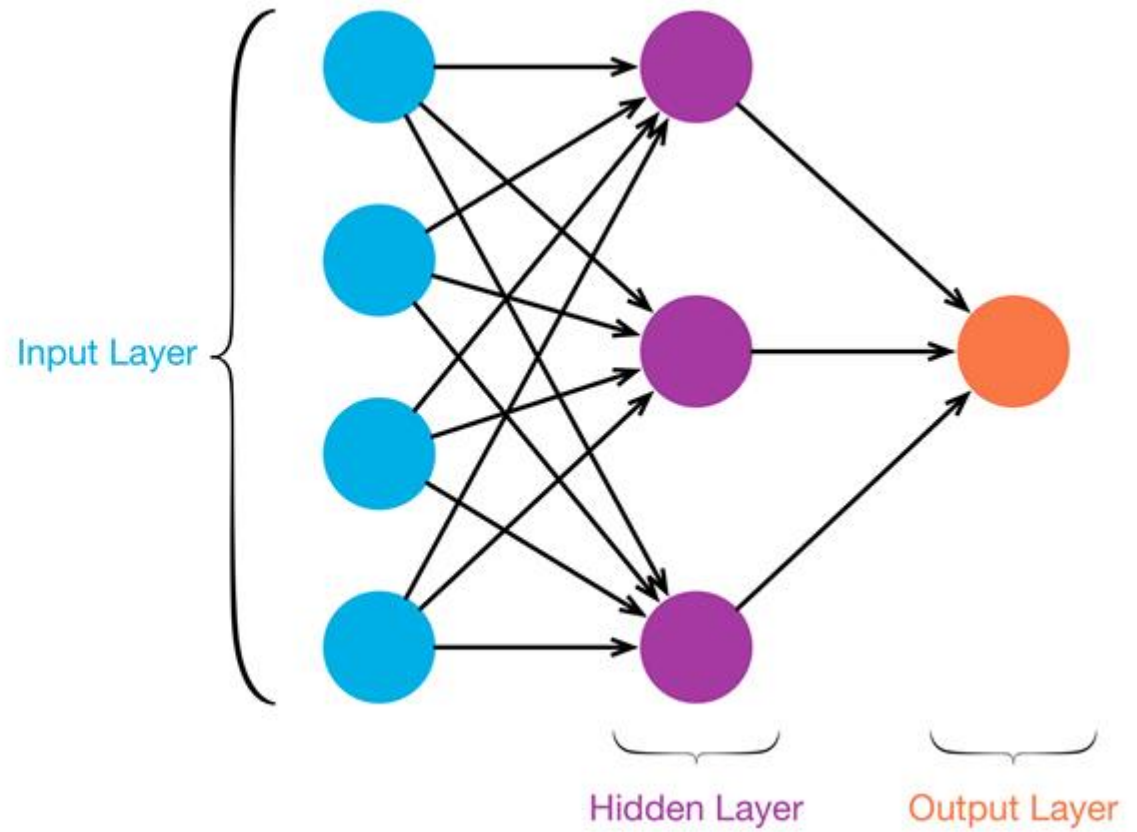
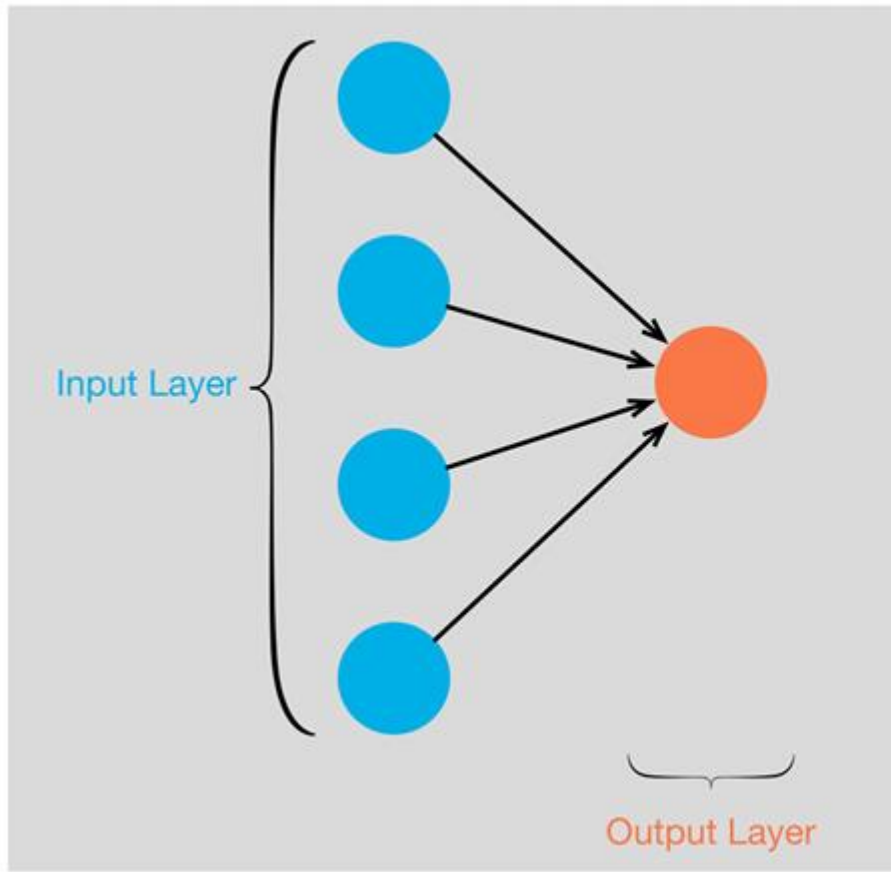


The Artificial Intelligence Perceptron

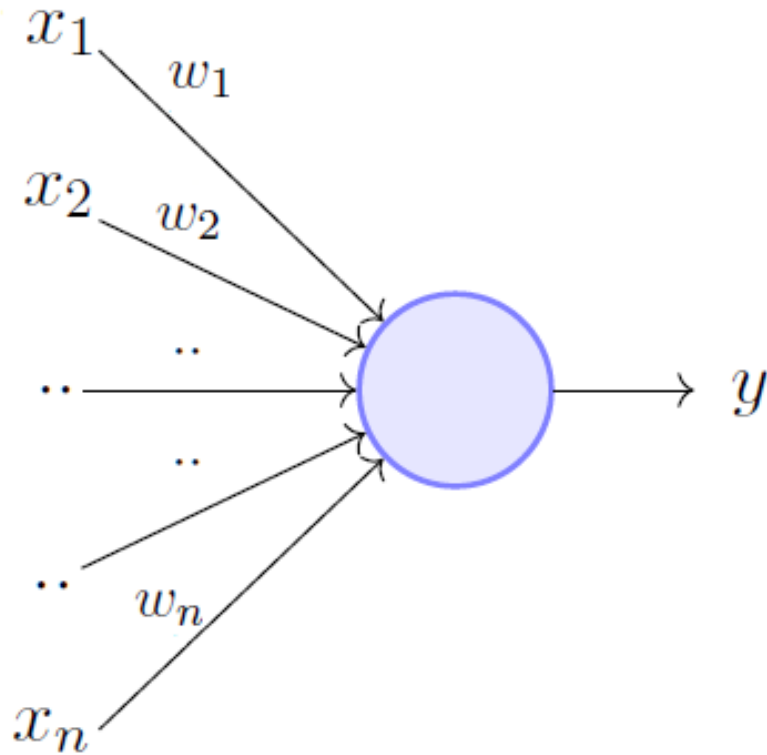


- **Single-Layer Perceptron:**
- This variant of perceptron is restricted to learning linearly separable patterns, making it suitable for tasks where data can be divided into distinct categories using a straight line.
- **Multilayer Perceptron:** Multilayer perceptron exhibit superior processing abilities due to their composition of two or more layers, enabling them to effectively manage more complex patterns and relationships present in the data.

Types of Perceptron



The Artificial Intelligence Perceptron



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

Rewriting the above,

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

Basic Components of Perceptron

- **Input Nodes or Input Layer**
 - input features (characteristics, or attributes)
- **Wight and Bias:**
 - Weight parameter represents the importance of the feature.
 - These weights are adjusted to learn the optimal values.
 - Bias can be considered as the line of intercept in a linear equation.
- **Summation Function**
 - The perceptron calculates the weighted sum of its inputs using the summation function.
 - The summation function combines the inputs with their respective weights to produce a weighted sum.

- **Activation Function**

- take the summed values as input and compare with the threshold and provide the output as 0 or 1.

- **Output**

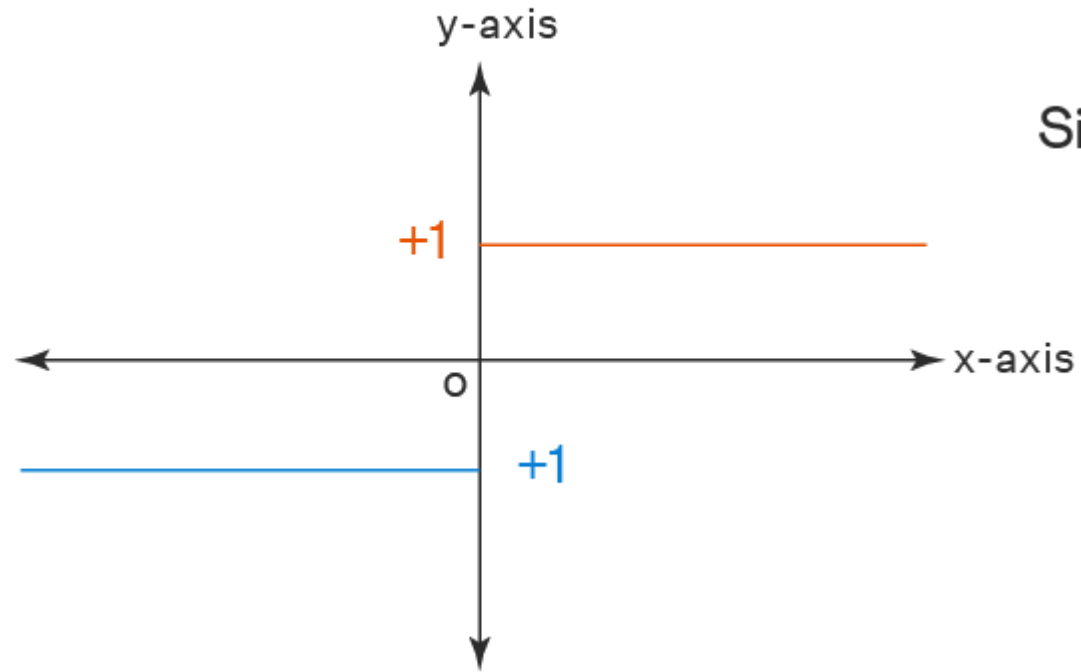
- The final output of the perceptron, is determined by the activation function's result.

- **Learning Algorithm (Weight Update Rule)**

- Throughout the training process, the perceptron undergoes learning by modifying its weights and bias, guided by a designated learning algorithm.
- One prevalent method is the perceptron learning algorithm, which iteratively adjusts the weights in response to the disparity between the predicted output and the actual output.

- **Role:** Activation functions introduce non-linearity to neural networks, allowing them to model complex relationships and learn non-linear patterns in data.
- **Output Transformation:** They transform the weighted sum of inputs of a neuron into its output.
- **Types:** Common activation functions include signum function, step function, and sigmoid function, tanh (hyperbolic tangent), ReLU (Rectified Linear Unit), and softmax.
- **Selection:** The choice of activation function depends on factors such as the nature of the data, the architecture of the neural network, and the task being performed.

- Signum Function



Signum Function

$$\text{Sgm}(f(x)) = \begin{cases} +1 & \text{for } x > 0 \\ -1 & \text{for } x < 0 \\ 0 & \text{for } x = 0 \end{cases}$$

Activation Function

- Find the output for the following values of x, using the signum function

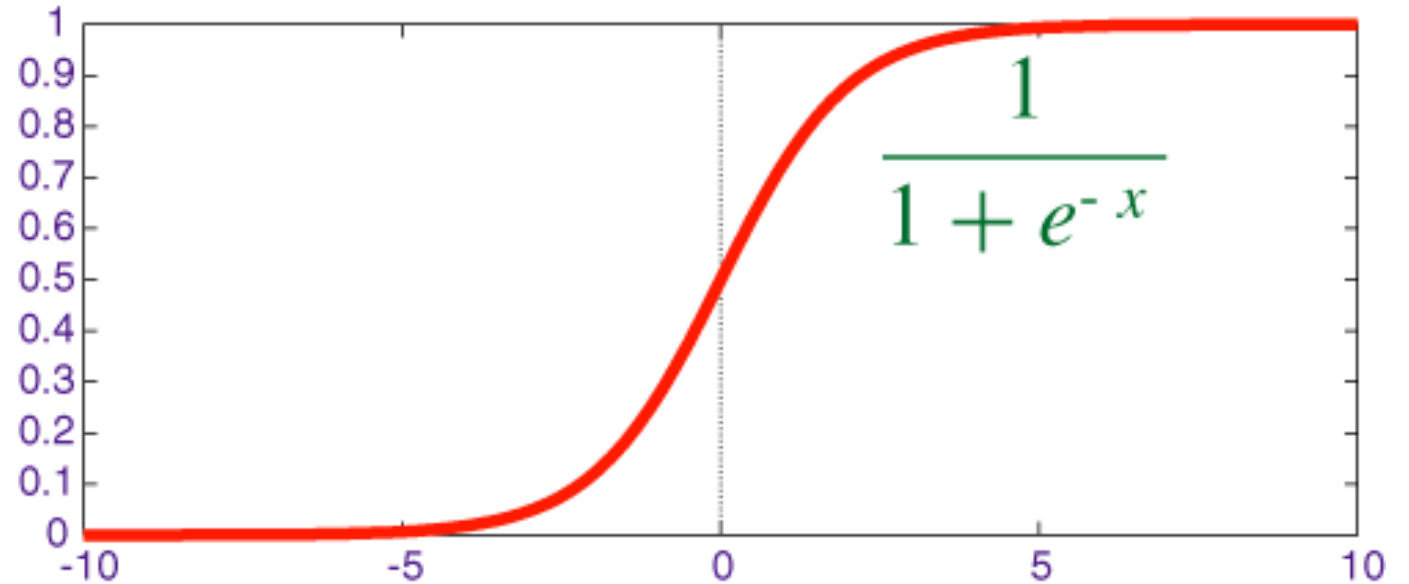
$$\text{Sgm}(f(x)) = \begin{cases} +1 & \text{for } x > 0 \\ -1 & \text{for } x < 0 \\ 0 & \text{for } x = 0 \end{cases}$$

$x = \{4.05, 17.60, 0, -11, 9.2, 3.99, -7.10\}$

Output = $\{+1, +1, 0, -1, +1, +1, -1\}$

Activation Function

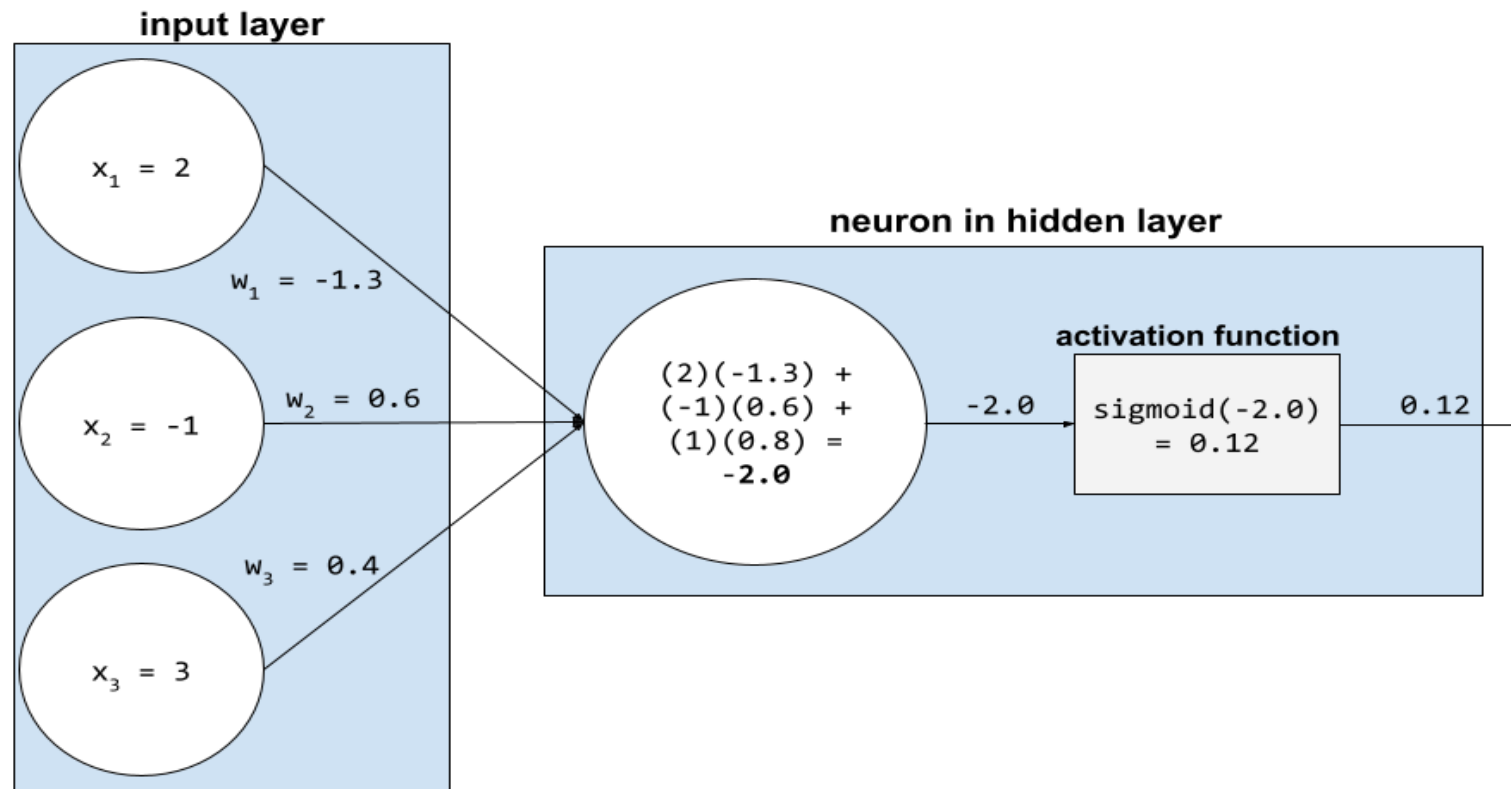
- the **Sigmoid Function** returns a value between 1 and 0, this is too powerful for Binary Classification Problems.
- It maps any input value into a range between 0 and 1, providing a way to normalize or 'squash' numbers.



- **e**: Is the Number of Euler
- **x**: Can be the output of your perceptron (**Neural Network**)

Activation Function

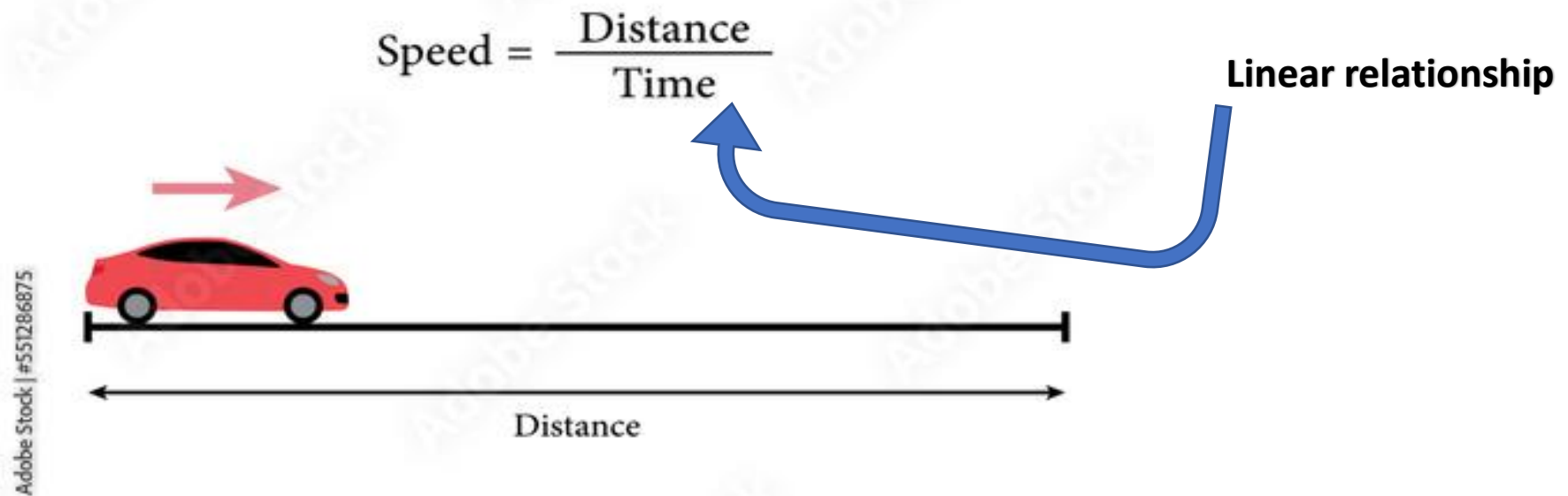
- For example, suppose the relevant input to a perceptron (neuron) consists of the following:



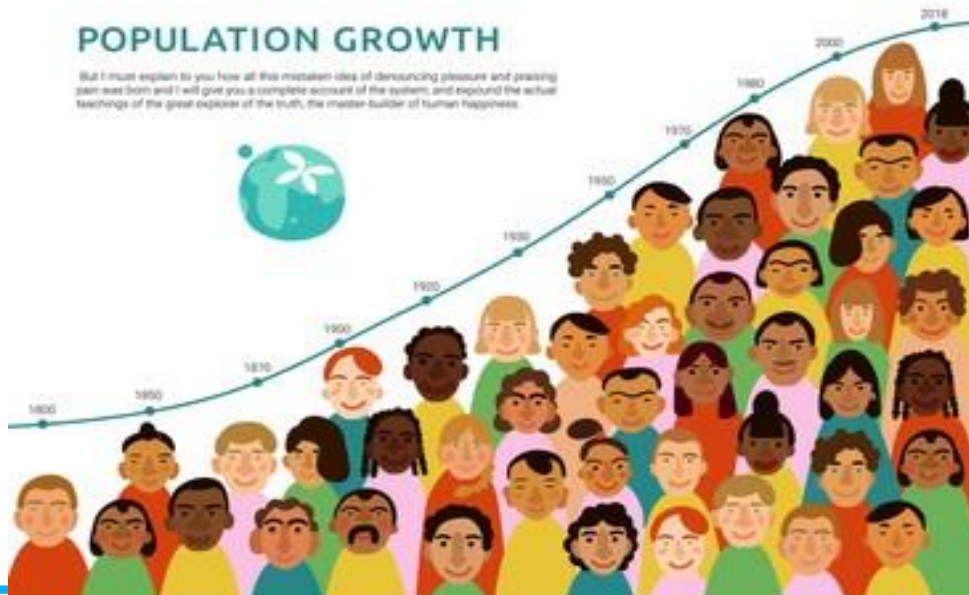
input value	input weight
2	-1.3
-1	0.6
3	0.4

Linear relationship vs Non-linear

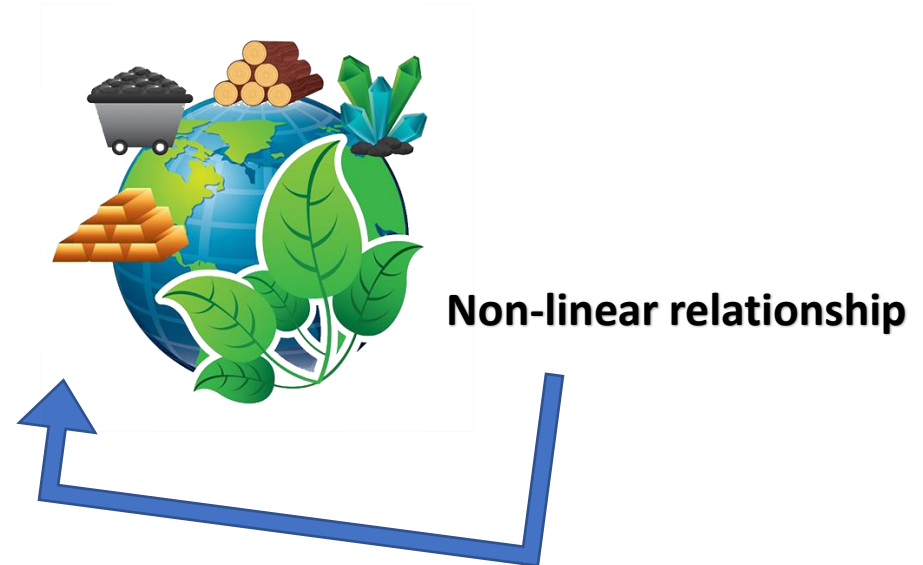
- In a nonlinear relationship, changes in the output do not change in direct proportion to changes in any of the inputs.
- A linear relationship creates a straight line when plotted on a graph.
- A nonlinear relationship does not create a straight line but instead creates a curve.



Linear relationship vs Non-linear



Natural Resources



How does Perceptron work?

- Randomly initialize the weights and biases (create a random hyperplane)
- computes the weighted sum of its inputs as:

$$z = w_1x_1 + w_2x_2 + \dots + w_nx_n = X^TW$$

- The activation function (**threshold logic units (TLU)**) compares this weighted sum to the threshold, which outputs 1 if the input is larger than a threshold value and 0 otherwise.

$$h(z) = \begin{cases} 0 & \text{if } z < Threshold \\ 1 & \text{if } z \geq Threshold \end{cases}$$

How does Perceptron work? (continue)

- During training, The perceptron's weights are adjusted to minimize the difference between the predicted output and the actual output.

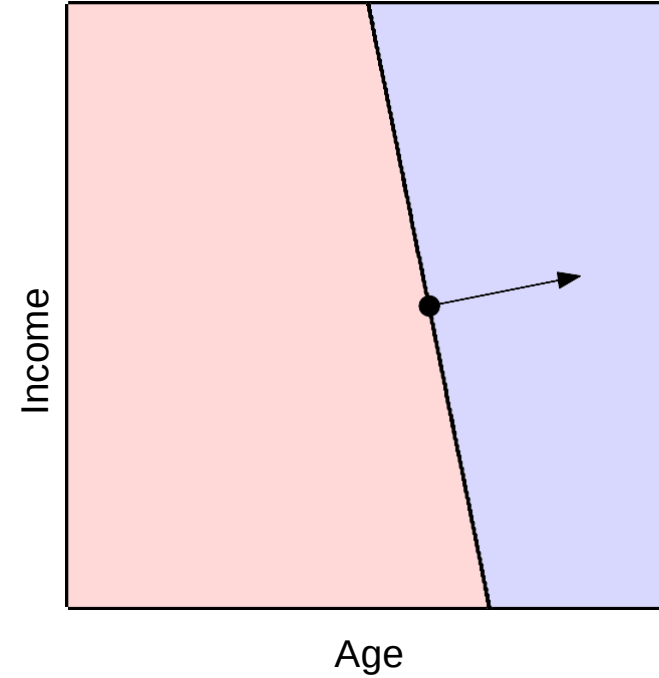
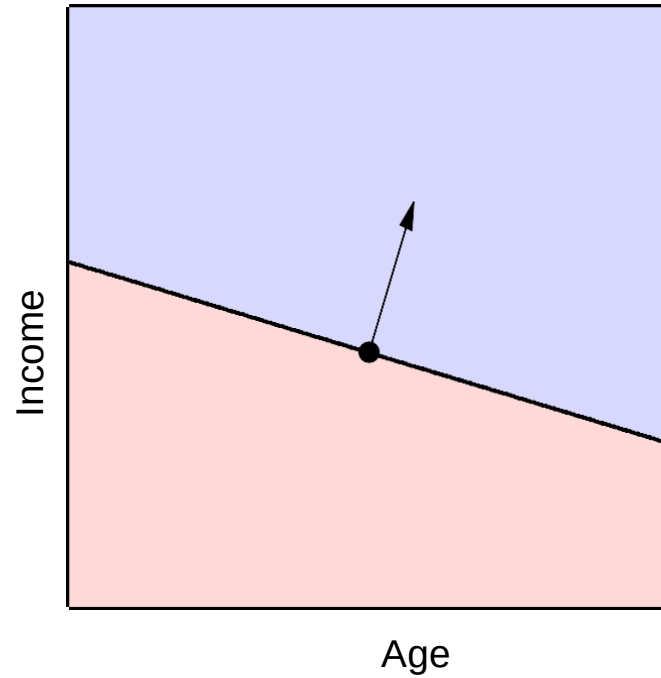
$$w_{i,j} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i$$

the learning rate.

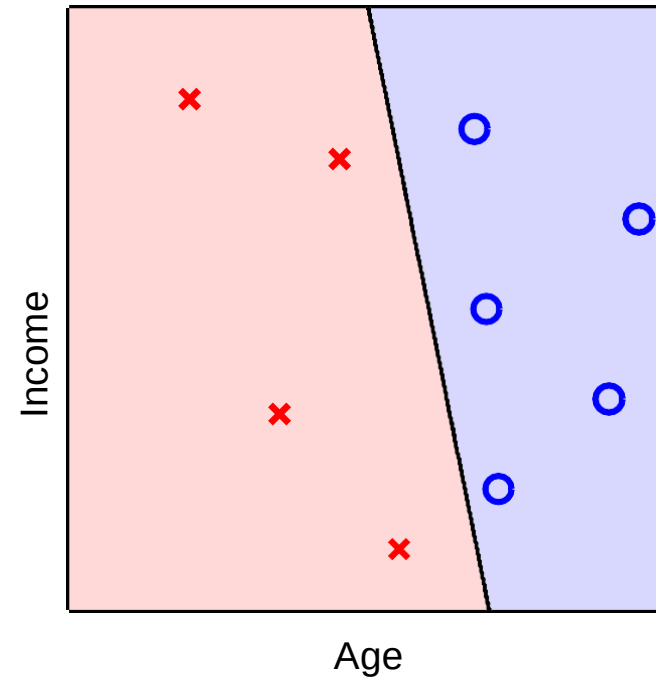
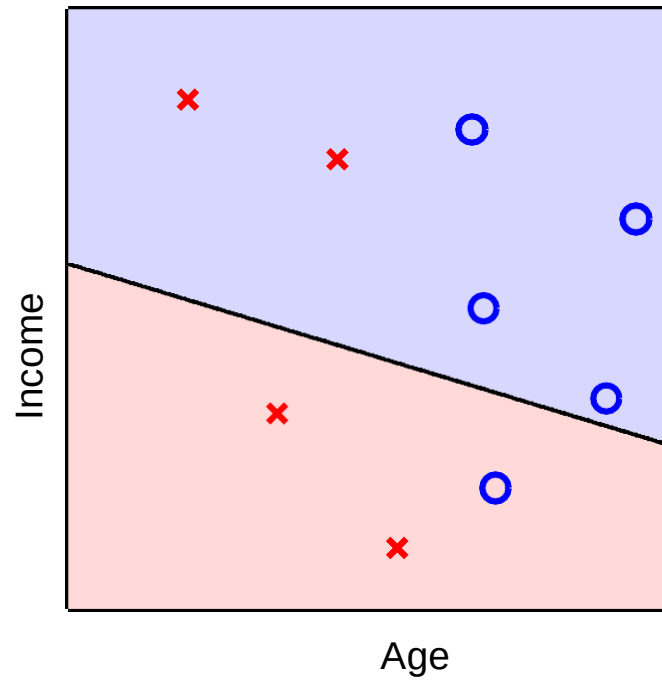
Actual output

Predicted output

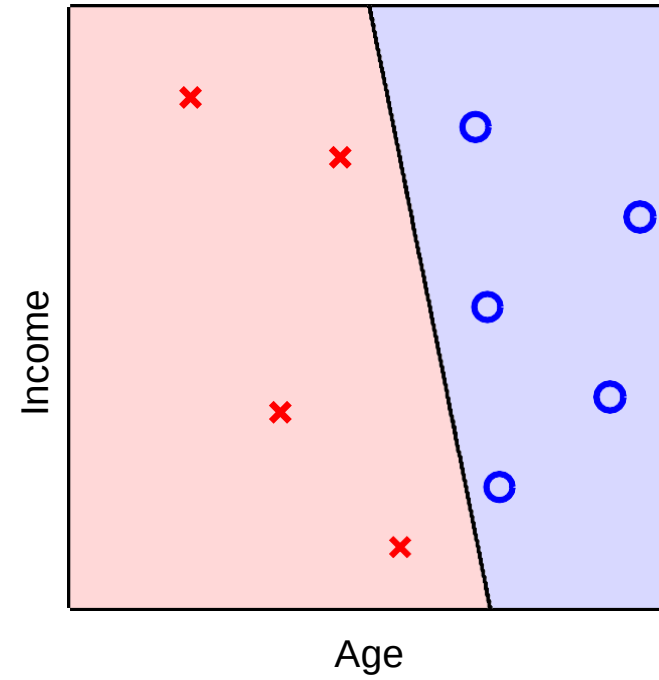
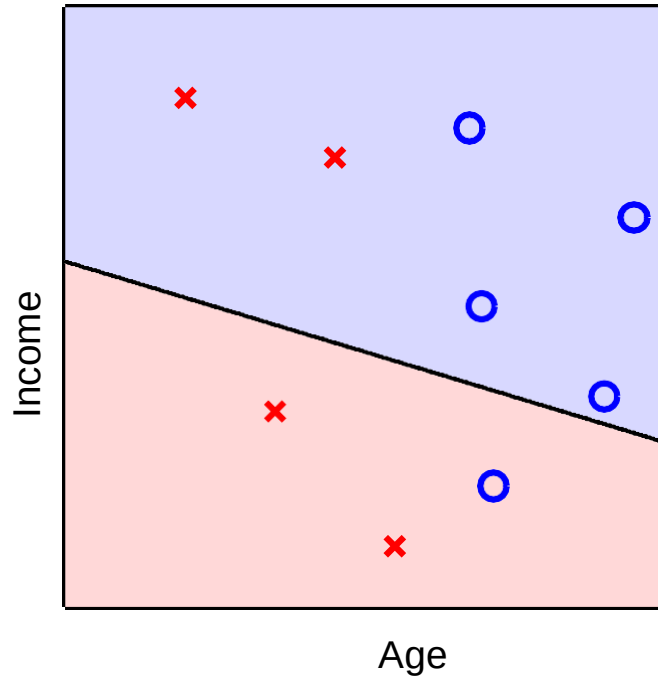
How does Perceptron work? (continue)



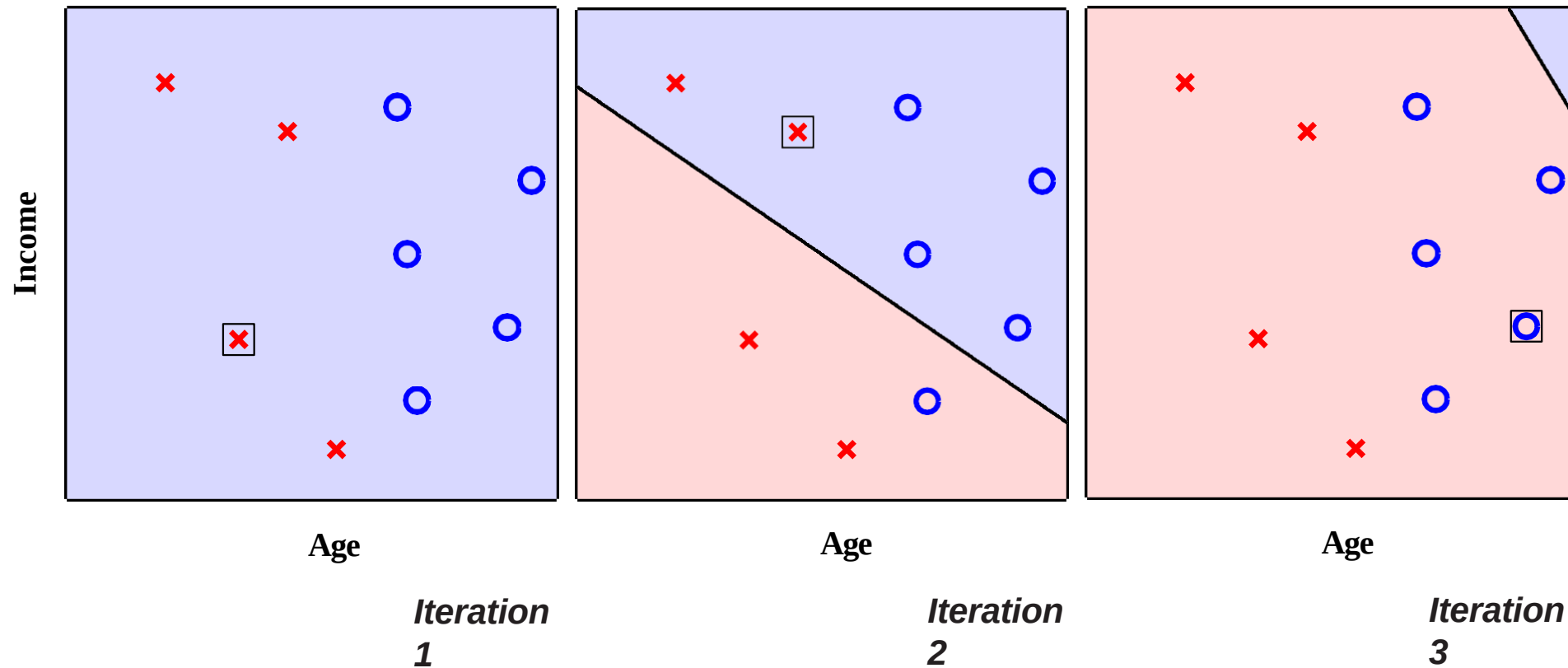
How does Perceptron work? (continue)



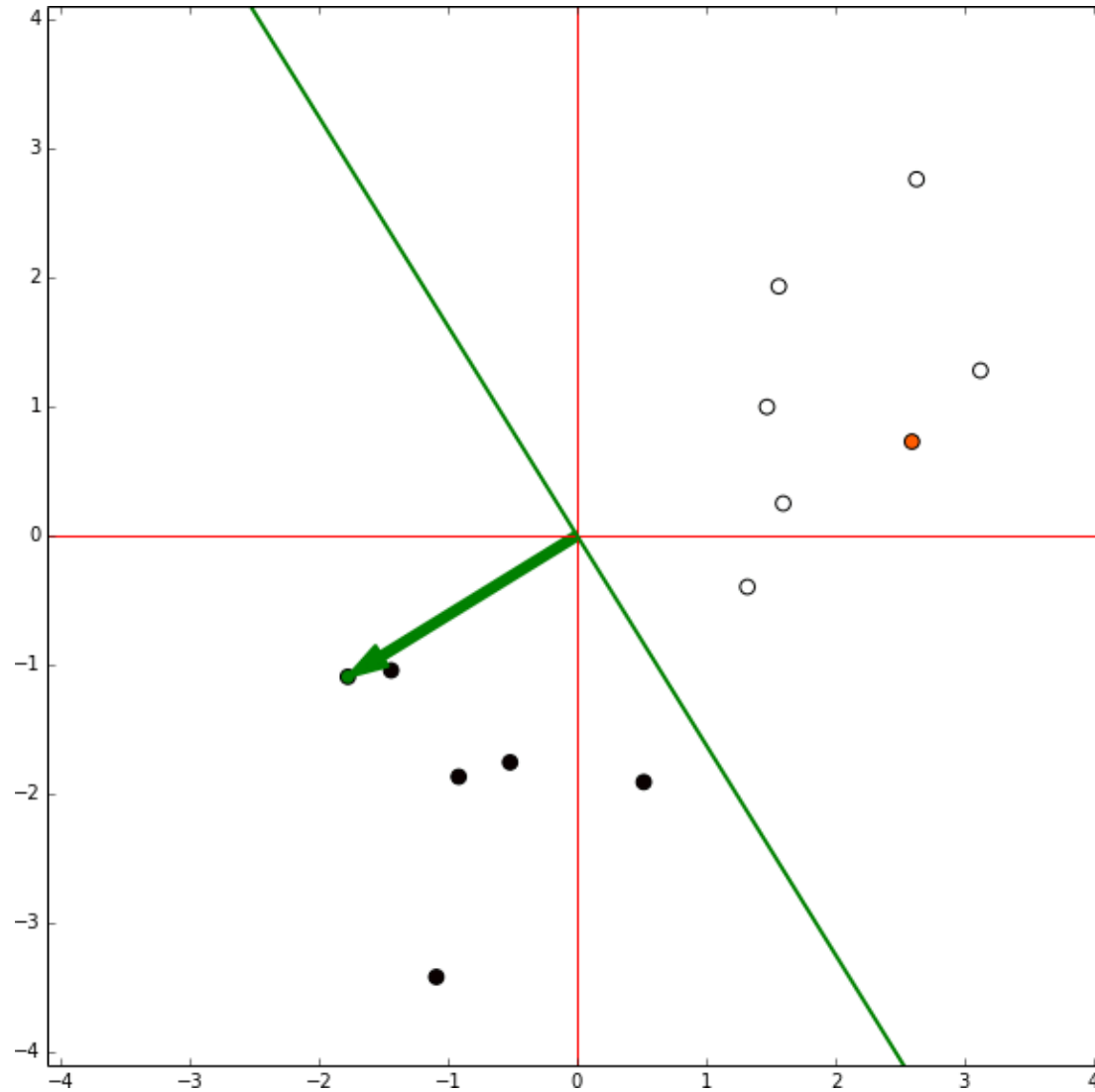
How does Perceptron work? (continue)



How does Perceptron work? (continue)



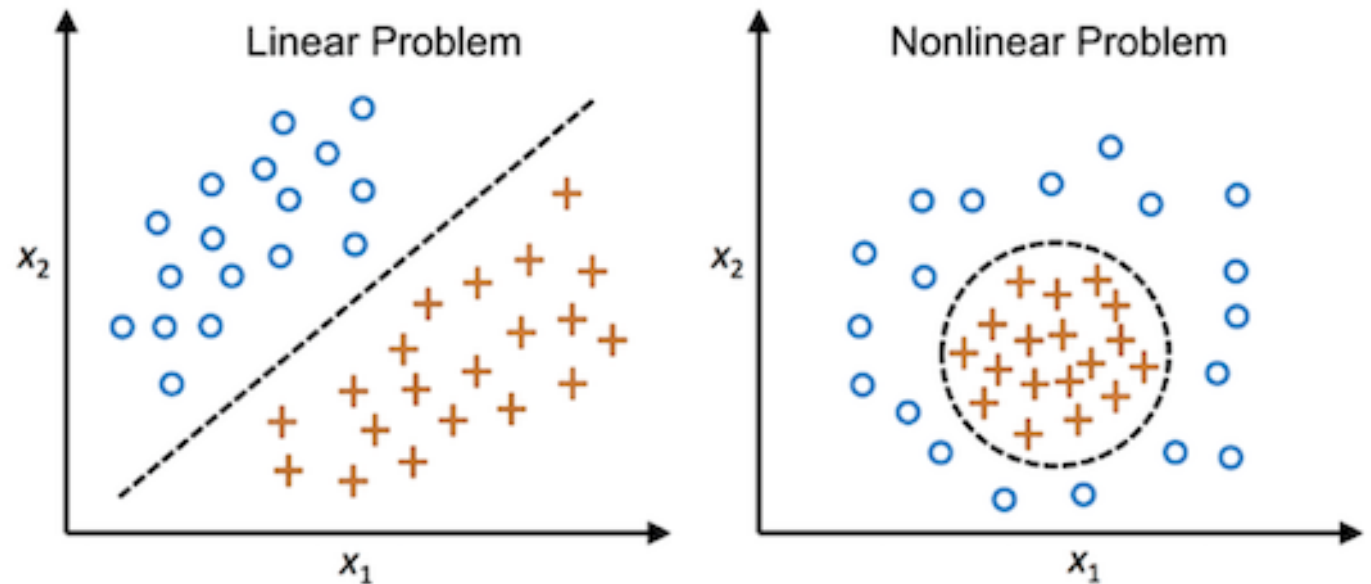
How does Perceptron work? (continue)



source: brilliant — [perceptron learning algorithm pg5](#)

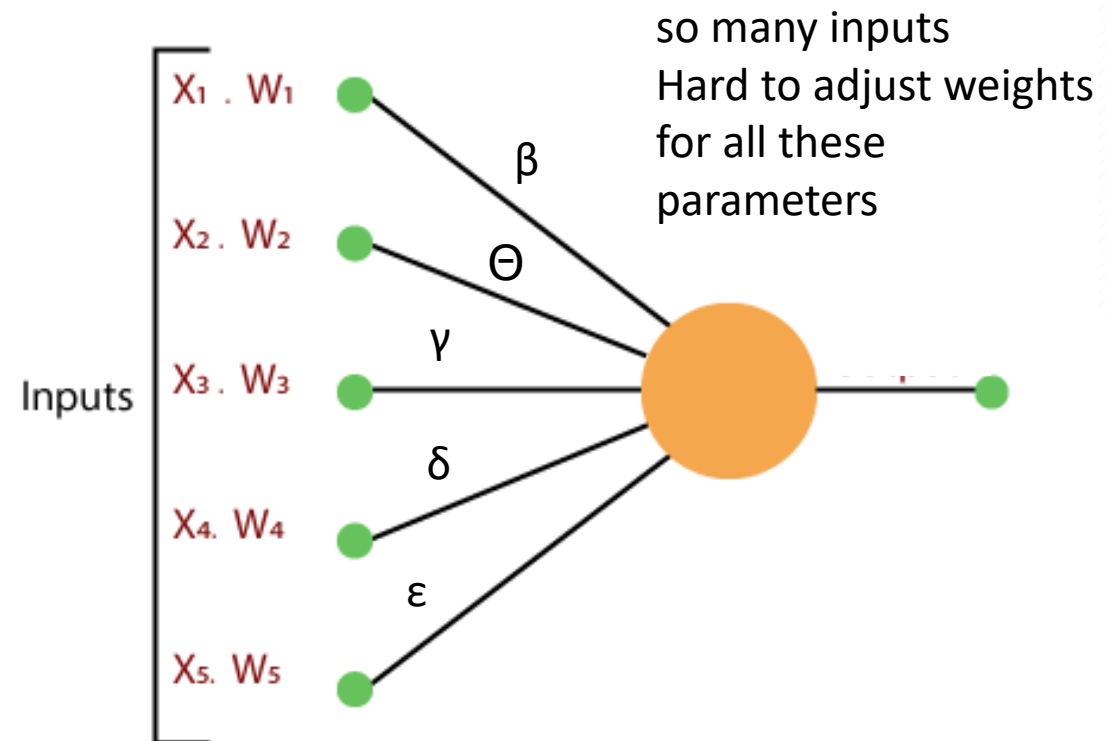
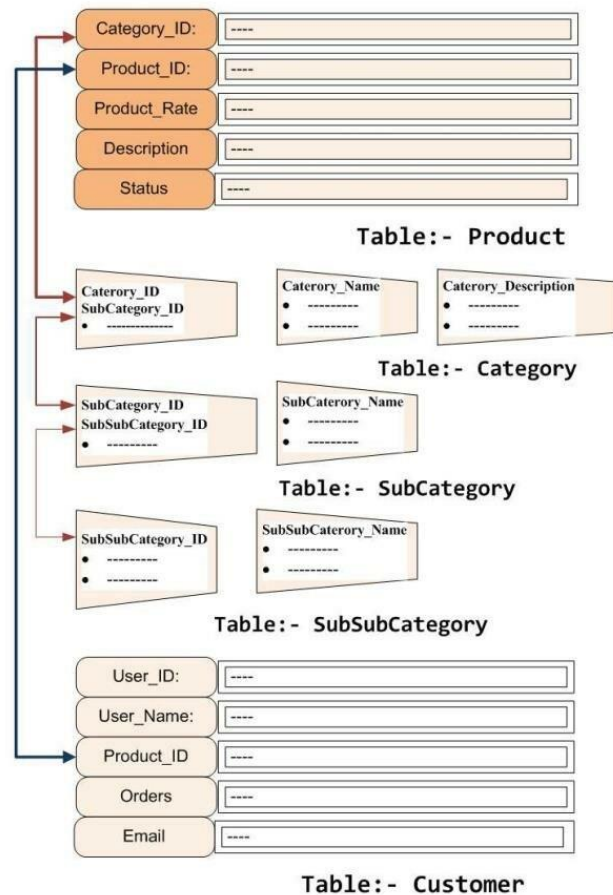
Limitations of Single-Layer Perceptron

- There are two major problems:
 - Single-Layer Perceptron's cannot classify non-linearly separable data points.
 - Complex problems, that involve a lot of features cannot be solved by Single-Layer Perceptron's.



Limitations of Single-Layer Perceptron

- Complex problems, that involve a lot of features cannot be solved by Single-Layer Perceptron's.

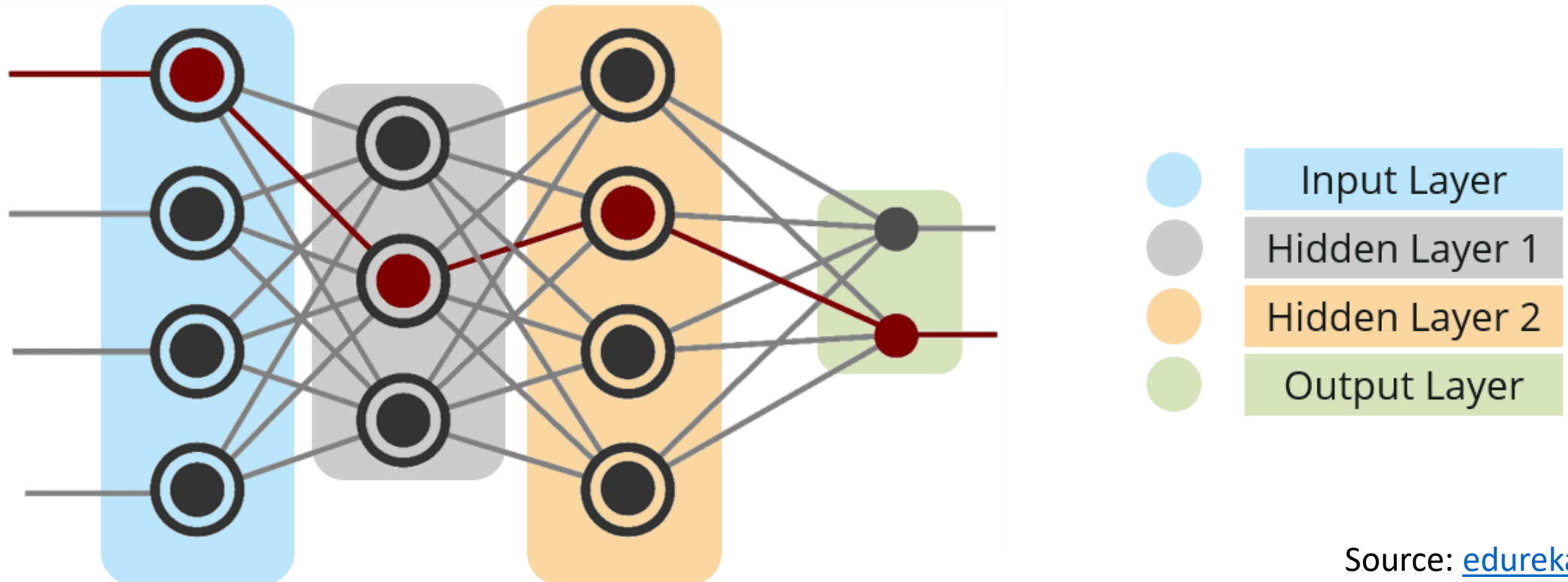


Applications of Single layer Perceptron

- Binary classification: One-layer neural networks can effectively classify binary datasets when the classes are linearly separable
- Feature extraction: One-layer neural networks can be employed as feature extractors in more complex models. By training a one-layer network on a specific task, the weights learned by the network can be used as features for subsequent machine learning algorithms. This technique is particularly useful when dealing with high-dimensional data, and it helps reduce the dimensionality of the input space.
- Simple pattern recognition: One-layer neural networks can recognize simple patterns and make basic decisions based on input features. For example, they can be used in simple image recognition tasks where the patterns to be recognized are linearly separable, such as detecting straight lines in images or distinguishing between basic shapes.
- Anomaly detection: One-layer neural networks can be utilized for detecting anomalies or outliers in datasets. If the majority of the data follows a linear pattern, a one-layer network can be trained to recognize that pattern. Any data point that deviates significantly from the learned linear boundary can be considered an anomaly.

What is Multi-Layer Perceptron?

- The brain is made up of millions of neurons, so a Neural Network is really just a composition of Perceptron's, connected in different ways and operating on different activation functions.



What is Multi-Layer Perceptron?

- **Input Nodes:** These nodes solely relay information to the hidden nodes without performing any computations themselves.
- **Hidden Nodes:** Hidden nodes, appropriately named for their lack of direct interaction with external data, serve as computational units within the network. They process information received from the input nodes and facilitate its transfer towards the output nodes. The collections of hidden nodes forms a "Hidden Layer." While each network includes a single input layer and output layer, it may incorporate zero or multiple hidden layers, with a Multi-Layer Perceptron featuring one or more hidden layers.
- **Output Nodes:** The Output nodes, constituting the "Output Layer," carry out computations and transmit processed information from the network to the external environment. These nodes are responsible for delivering the network's output to the outside world.

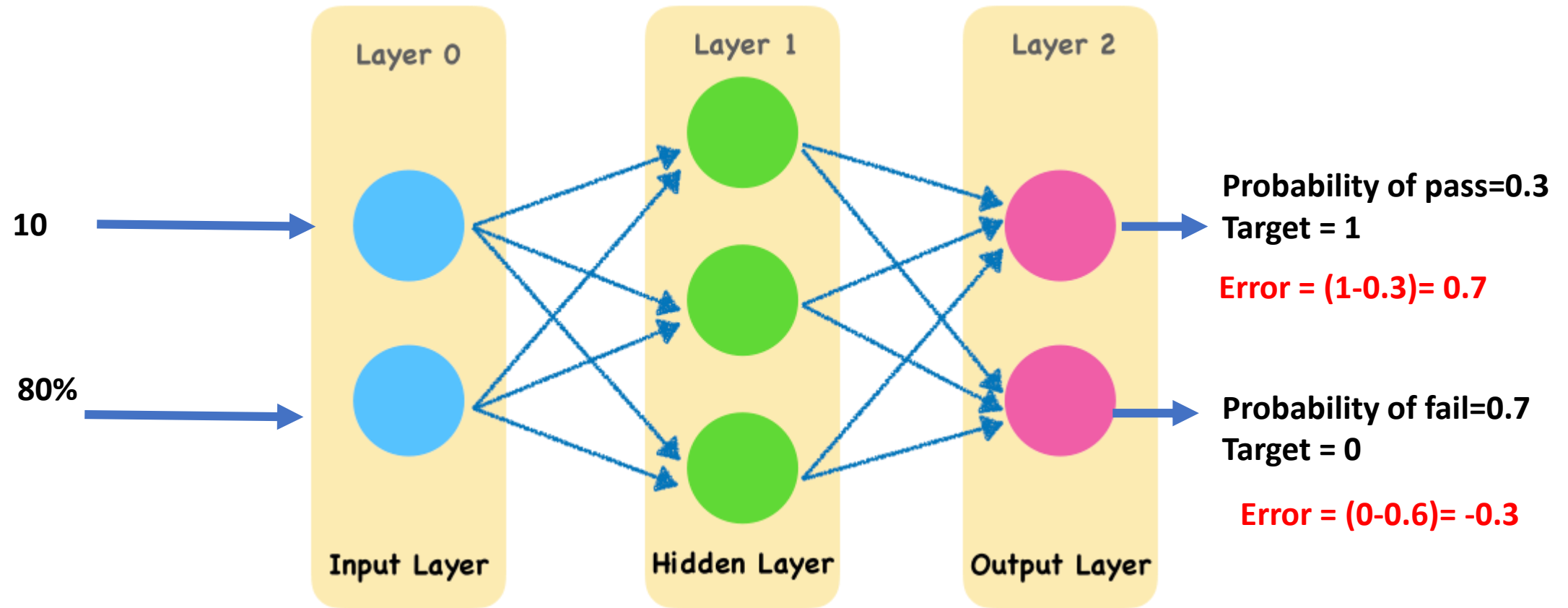
Multi-Layer Perceptron Example

Consider a dataset representing student performance in an exam. The dataset consists of three columns: the first column represents the number of hours spent studying, the second column indicates the concentration within the class, and the third column denotes whether the student passed or failed the final exam. For instance, if a student studied for 10 hours, and his concentration an average of 80% in class, and ultimately passed the final exam, the corresponding entry in the dataset would be [10, 80, Passed].

Can you predict whether student will pass the semester or not, if the he studied 2 hrs. and the his concentration is 38%.

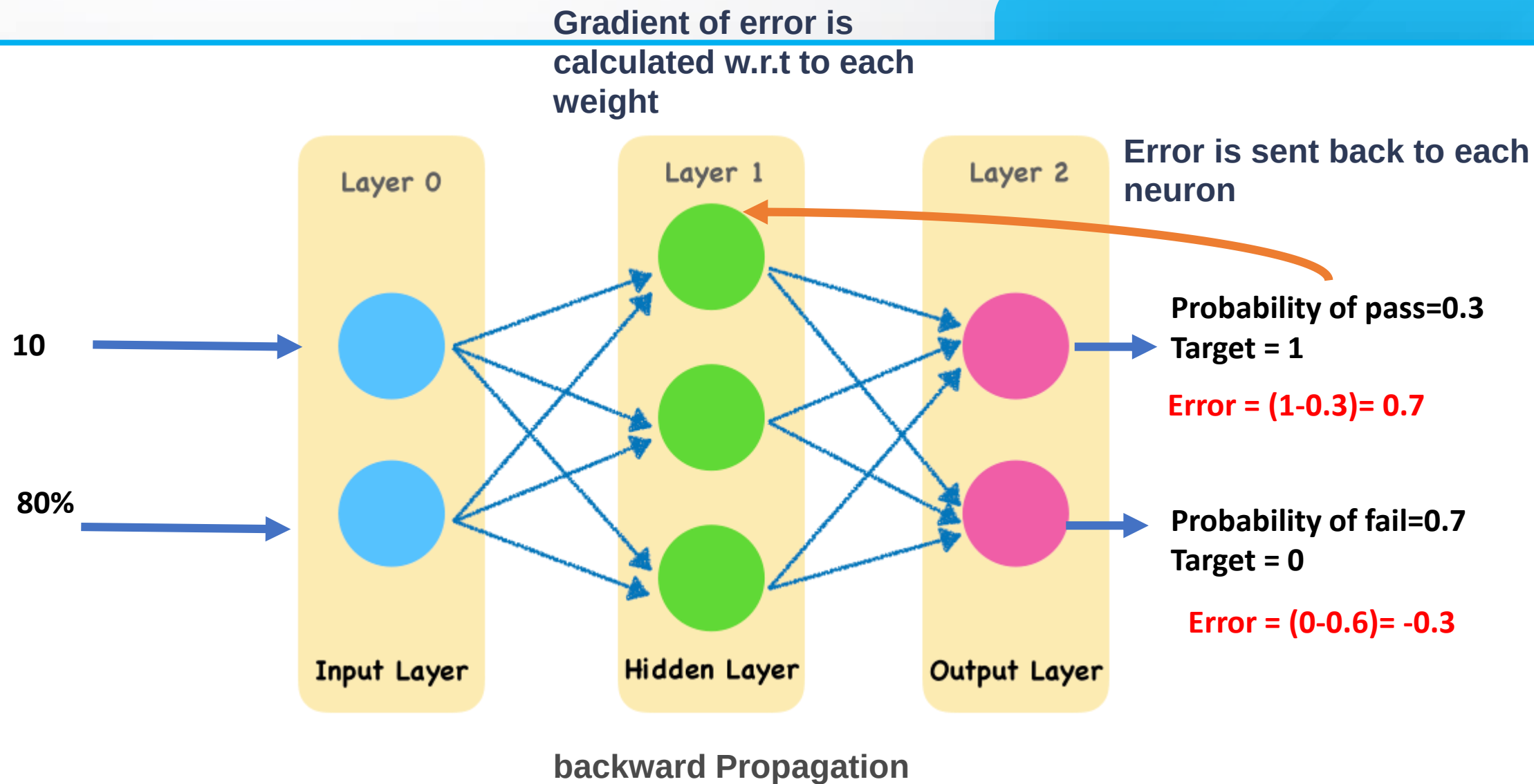
# of hours spent studying	concentration within the class	passed or failed
4	60%	pass (1)
10	80%	pass (1)
8	20%	fail (0)
6	10%	pass (1)
4	95%	fail (0)

Multi-Layer Perceptron Example



Forward Propagation

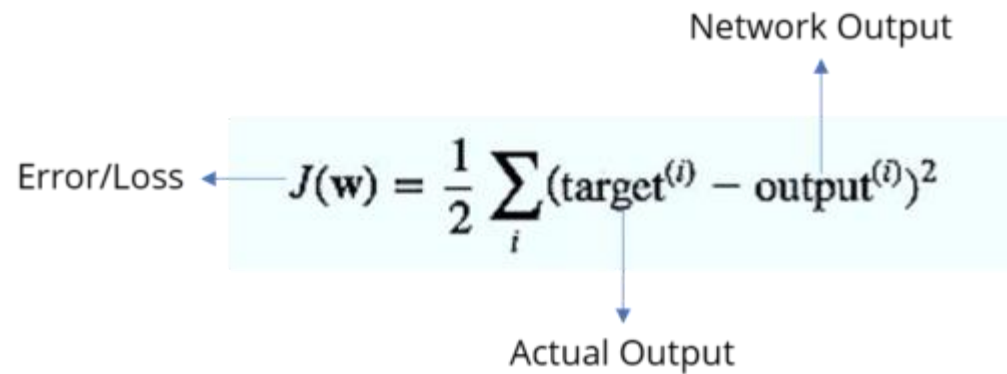
Multi-Layer Perceptron Example



Multi-Layer Perceptron Example

Based on the error we got, it will calculate the rate of change of error w.r.t change in the weights.

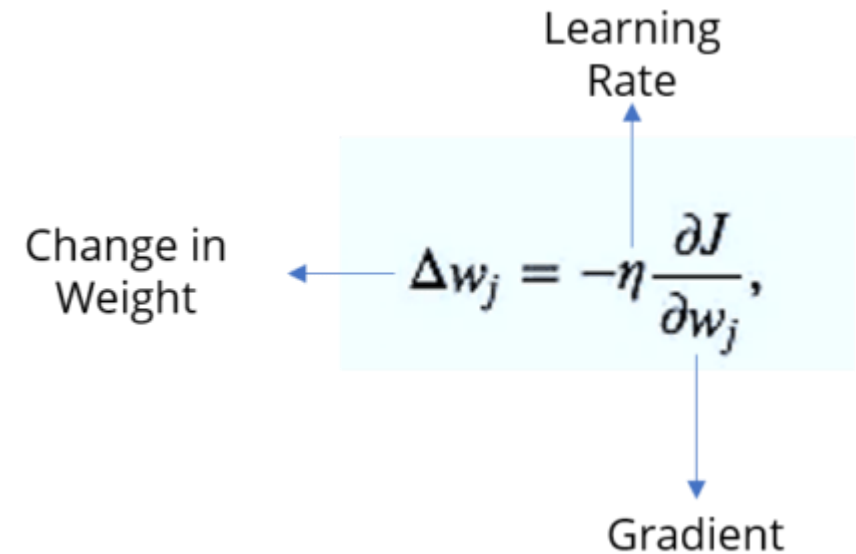
First we calculate the error, consider the equation below:



The diagram shows the equation $J(\mathbf{w}) = \frac{1}{2} \sum_i (\text{target}^{(i)} - \text{output}^{(i)})^2$ enclosed in a light blue box. Three arrows point from the equation to labels: an arrow from the left points to 'Error/Loss', an arrow from the 'output' term points down to 'Actual Output', and an arrow from the 'output' term points up to 'Network Output'.

$$J(\mathbf{w}) = \frac{1}{2} \sum_i (\text{target}^{(i)} - \text{output}^{(i)})^2$$

Labels: Error/Loss, Network Output, Actual Output

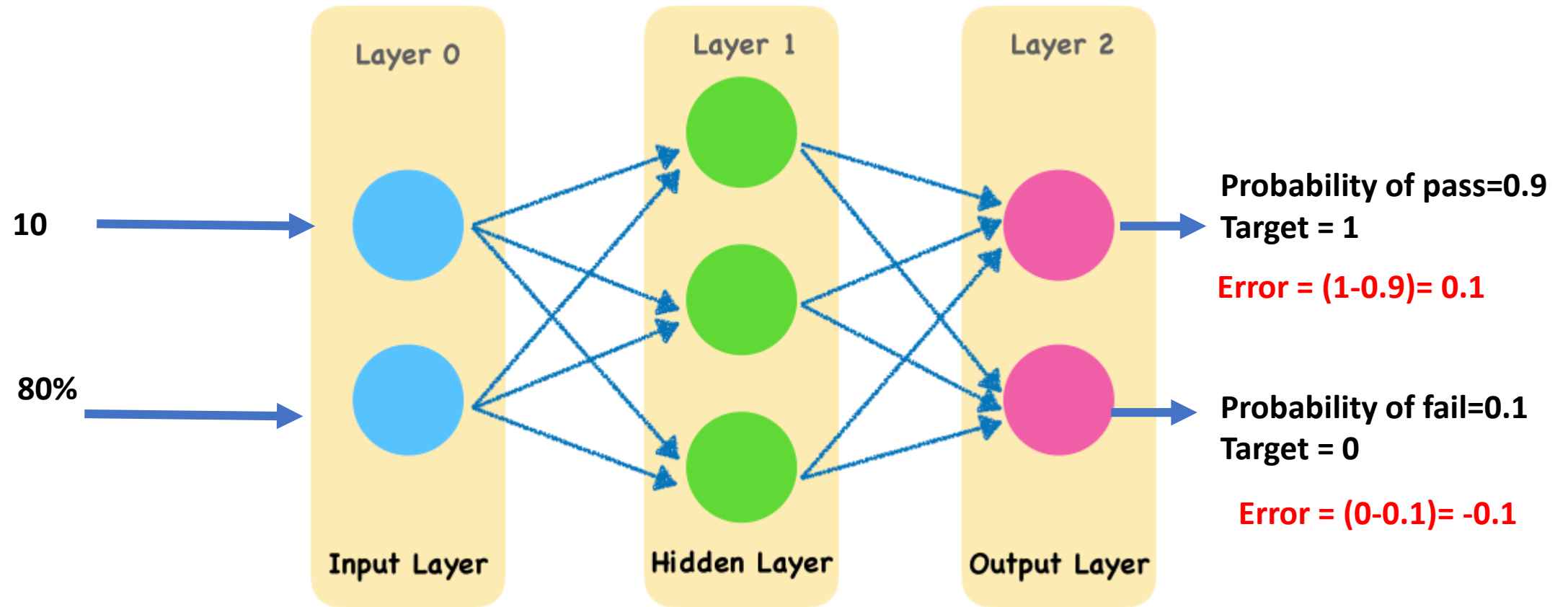


The diagram shows the equation $\Delta w_j = -\eta \frac{\partial J}{\partial w_j}$ enclosed in a light blue box. Three arrows point from the equation to labels: an arrow from the left points to 'Change in Weight', an arrow from the top points to 'Learning Rate', and an arrow from the bottom points to 'Gradient'.

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j}$$

Labels: Change in Weight, Learning Rate, Gradient

Multi-Layer Perceptron Example



Forward Propagation

Example

- Suppose you have:
- Input $x_1=0.5$, $x_2=1.0$
- Weights $w_1=0.4$, $w_2=0.6$
- Learning rate $\eta=0.1$
- Target output $y=1$
- Activation function $\phi(z)$ is the sigmoid function, where $\phi(z) = \frac{1}{1+e^{-z}}$

Example

1- Compute the weighted sum:

- $z = (0.4 \times 0.5) + (0.6 \times 1.0) = 0.2 + 0.6 = 0.8$

2- Apply the activation function (sigmoid in this case):

$$\hat{y} = \frac{1}{1 + e^{-0.8}} \approx 0.69$$

3- Calculate the error:

$$E = \frac{1}{2}(1 - 0.69)^2 \approx 0.048$$

4- Compute the gradient of the error with respect to w1:

$$\frac{\partial E}{\partial w_i} = -(y - \hat{y}) \cdot \phi'(z) \cdot x_i$$

$$\frac{\partial E}{\partial w_1} = -(1 - 0.69) \times \phi'(0.8) \times 0.5$$

Example

- For a sigmoid function, $\phi'(z) = \phi(z)(1 - \phi(z))$

$$\phi'(0.8) \approx 0.69 \times (1 - 0.69) = 0.214$$

$$\frac{\partial E}{\partial w_1} \approx -0.31 \times 0.214 \times 0.5 \approx -0.033$$

5- Update the weight w_1 :

$$\Delta w_1 = 0.1 \times -0.033 = -0.0033$$

$$w_1 (\text{new}) = 0.4 + (-0.0033) = 0.3967$$

Progression to DEEP (Linear) Neural Networks

Single-layer:

$$h = \mathbf{w}^T \mathbf{x}$$

Hidden-layer:

$$\mathbf{h} = \mathbf{W}^T \mathbf{x}$$

Two Hidden Layers:

$$\mathbf{h} = \mathbf{W}_2^T \mathbf{W}_1^T \mathbf{x}$$

Three Hidden Layers:

$$\mathbf{h} = \mathbf{W}_3^T \mathbf{W}_2^T \mathbf{W}_1^T \mathbf{x}$$

A LOT of Hidden Layers:

$$\mathbf{h} = \mathbf{W}_N^T \dots \mathbf{W}_2^T \mathbf{W}_1^T \mathbf{x}$$



Bird Beak? **Yes**



Bird Head?
Yes



Bird Eyes? **Yes**



Bird leg? **Yes**



Bird Body?
Yes



Bird Tail? **Yes**



Is it Bird? **Yes**

Practice



- Use Multilayer perceptron to predict if the bank note is real or fake, use the following dataset:

4.0127	10.1477	-3.9366	-4.0728	0
2.6606	3.1681	1.9619	0.18662	0
3.931	1.8541	-0.02343	1.2314	0
0.01727	8.693	1.3989	-3.9668	0
3.2414	0.40971	1.4015	1.1952	0
2.2504	3.5757	0.35273	0.2836	0
-1.3971	3.3191	-1.3927	-1.9948	1
0.39012	-0.14279	-0.03199	0.35084	1
-1.6677	-7.1535	7.8929	0.96765	1
-3.8483	-12.8047	15.6824	-1.281	1
-3.5681	-8.213	10.083	0.96765	1
-2.2804	-0.30626	1.3347	1.3763	1
-1.7582	2.7397	-2.5323	-2.234	1
-0.89409	3.1991	-1.8219	-2.9452	1
0.3434	0.12415	-0.28733	0.14654	1

Features

- Variance of Wavelet Transformed image
- Skewness of Wavelet Transformed image
- Curtosis of Wavelet Transformed image
- Entropy of image

Label

1 – Real, 0 - Fake

Additional Resources

- ECE595ML Lecture 16-1 Perceptron
 - <https://www.youtube.com/watch?v=amOdxJ6TGWg&list=PL4FSfq6xtSvyqEsz3UUnAizemXJfQyDVD&index=36>



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY



Thank You

