



TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC TP. HỒ CHÍ MINH
HO CHI MINH CITY UNIVERSITY OF FOREIGN LANGUAGES - INFORMATION TECHNOLOGY

Mạng nơron nhân tạo

Biên soạn: ThS. Vũ Đình Ái (aivd@huflit.edu.vn)

Cập nhật: tháng 09/2023

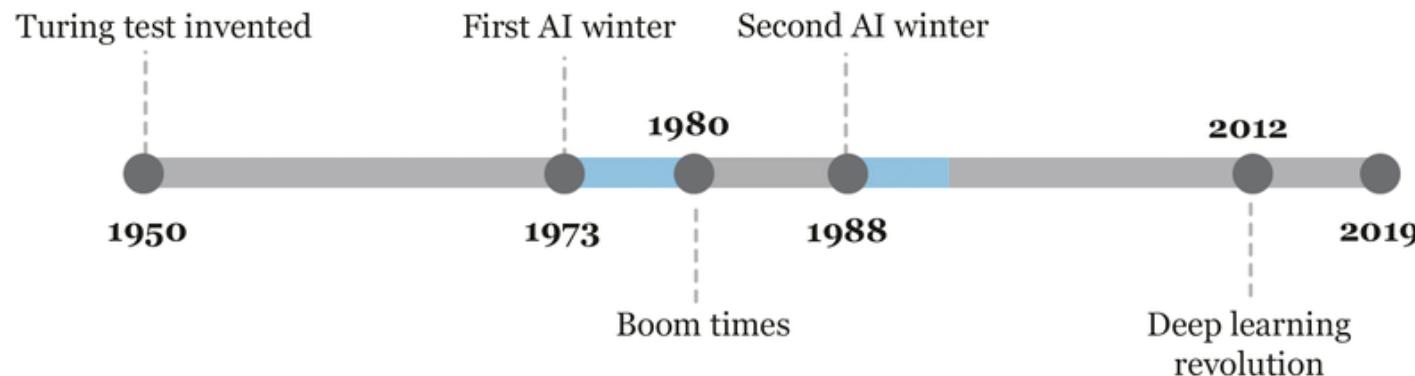
www.huflit.edu.vn

Nội dung

- Giới thiệu mạng nơron nhân tạo
- Ứng dụng của mạng nơron
- Cấu tạo nơron nhân tạo
- Kiến trúc mạng nơron
- Thuật toán PLA
- Thuật toán MLPs

Giới thiệu mạng nơron nhân tạo

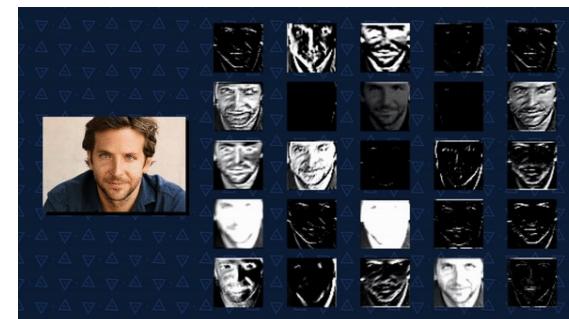
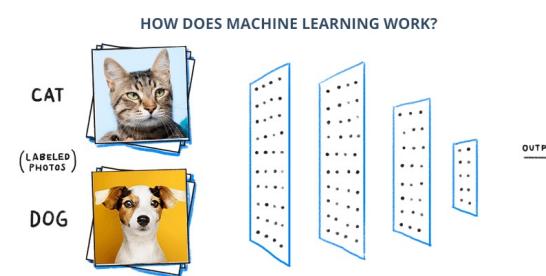
- Mạng nơron nhân tạo (artificial neural network - ANN) trong học máy được đề xuất đầu tiên bởi Warren McCulloch và Walter Pitts vào năm 1943. Trong bài báo nổi tiếng của họ, "A Logical Calculus of Ideas Immanent in Nervous Activity", McCulloch và Pitts đã mô tả một mô hình toán học của nơron sinh học, từ đó hình thành cơ sở cho mạng nơron nhân tạo.
- Mô hình này dựa trên ý tưởng rằng nơron có thể được biểu diễn như một hàm toán học đơn giản với đầu vào và đầu ra, và nhiều nơron có thể kết hợp với nhau để tạo thành một mạng lưới, giống như trong bộ não của con người.



Ứng dụng của mạng nơ-ron

▪ Thị giác máy tính

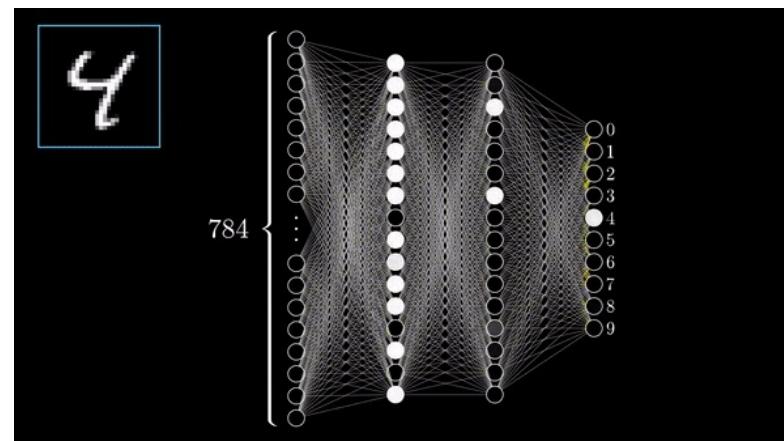
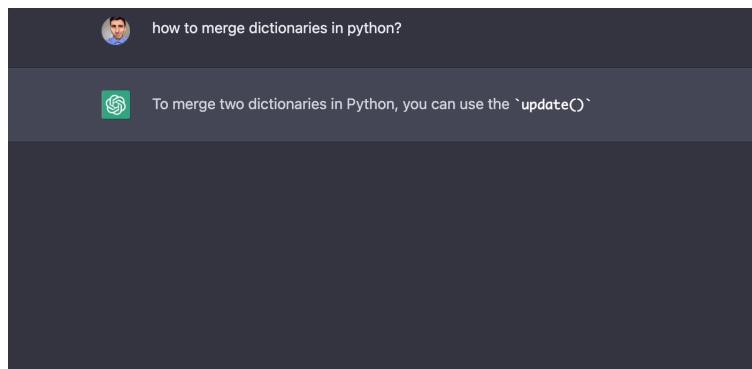
- Thị giác máy tính là khả năng trích xuất dữ liệu cũng như thông tin chuyên sâu từ hình ảnh và video của máy tính. Với mạng nơ-ron, máy tính có thể phân biệt và nhận diện hình ảnh tương tự như con người. Thị giác máy tính được ứng dụng trong nhiều trường hợp, chẳng hạn như:
 - ✓ Hệ thống nhận diện hình ảnh trên ô tô tự lái để chúng có thể nhận ra các biển báo giao thông cũng như những người tham gia giao thông khác
 - ✓ Kiểm duyệt nội dung để tự động loại bỏ nội dung không an toàn hoặc không phù hợp khỏi kho lưu trữ hình ảnh và video



Ứng dụng của mạng nơ-ron

▪ Kỹ thuật xử lý ngôn ngữ tự nhiên

- Kỹ thuật xử lý ngôn ngữ tự nhiên (NLP) là khả năng xử lý văn bản tự nhiên do con người tạo ra. Mạng nơ-ron giúp máy tính thu thập thông tin chuyên sâu và ý nghĩa từ dữ liệu văn bản và tài liệu. NLP được sử dụng trong nhiều trường hợp, bao gồm trong những chức năng sau:
 - ✓ Tổng đài viên ảo và chatbot tự động
 - ✓ Tự động sắp xếp và phân loại dữ liệu được ghi
 - ✓ Lập chỉ mục các cụm từ quan trọng thể hiện cảm xúc, ví dụ như những bình luận tích cực và tiêu cực trên mạng xã hội
 - ✓ Tóm tắt tài liệu và tạo bài viết về một chủ đề cho trước



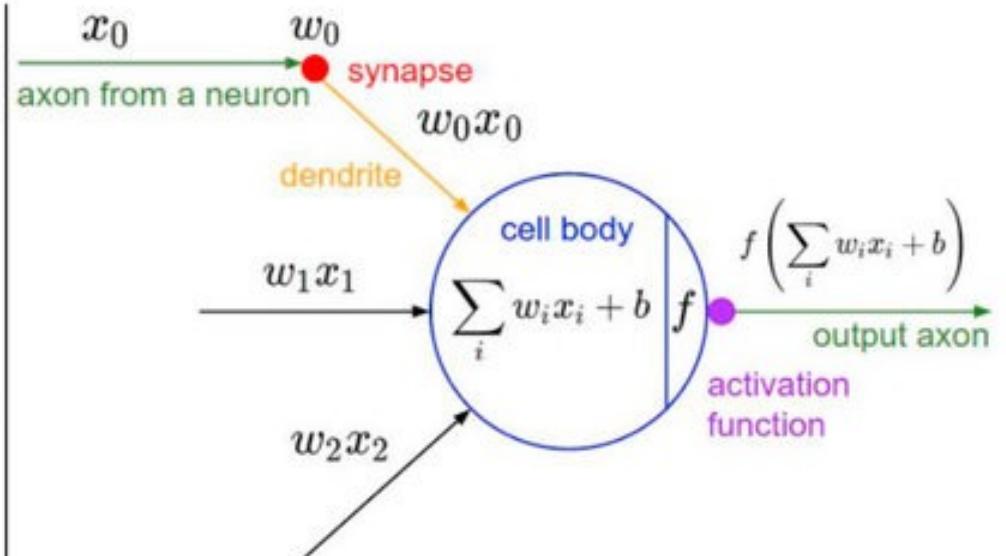
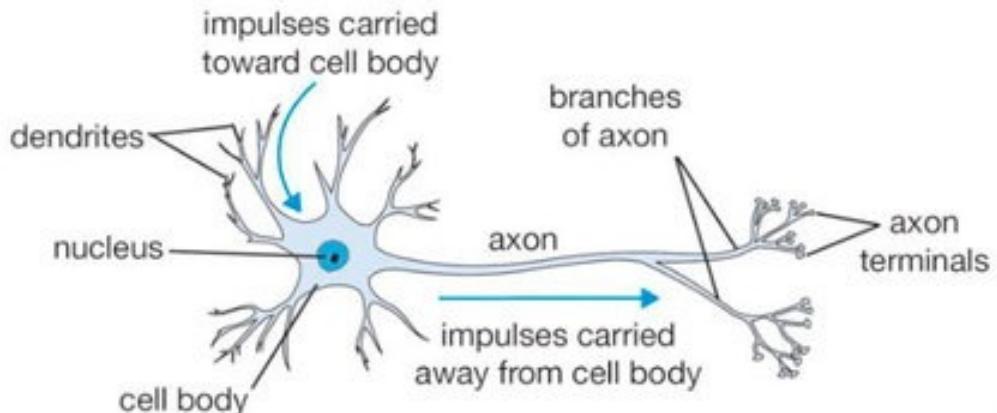
Cấu tạo nơron nhân tạo

- ANN có thể được xem như một cấu trúc xử lý thông tin một cách phân tán và song song ở mức cao
- ANN có khả năng học (learn), nhớ lại (recall), và khái quát hóa (generalize) từ các dữ liệu học
- Khả năng của một ANN phụ thuộc vào
 - Kiến trúc (topology) của mạng nơ-ron
 - Đặc tính đầu vào/ra của mỗi nơ-ron
 - Thuật toán học (huấn luyện)
 - Dữ liệu học

Giới thiệu mạng nơ-ron nhân tạo

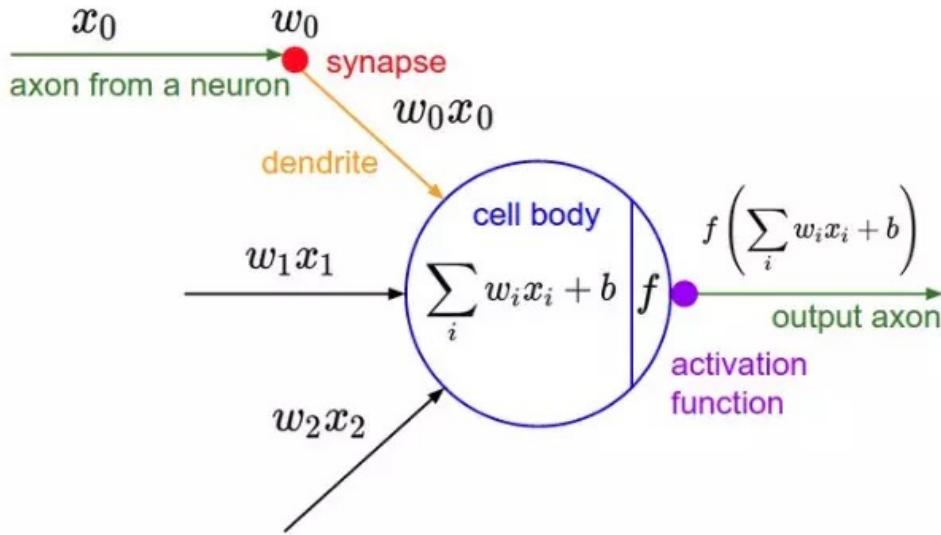
- Mạng nơ-ron nhân tạo (Artificial neural network – ANN)
 - Mô phỏng các hệ thống nơ-ron sinh học (các bộ não con người)
 - ANN là một cấu trúc (structure/network) được tạo nên bởi một số lượng các nơ-ron (artificial neurons) liên kết với nhau
- Mỗi nơ-ron
 - Có một đặc tính vào/ra
 - Thực hiện một tính toán cục bộ (một hàm cục bộ)
- Giá trị đầu ra của một nơ-ron được xác định bởi
 - Đặc tính vào/ra của nó
 - Các liên kết của nó với các nơ-ron khác
 - (Có thể) các đầu vào bổ sung

Cấu tạo nơron nhân tạo



Biological Neuron: Nơron sinh học	Artificial Neuron Nơron nhân tạo
Nucleus (cell body) : Nhân tế bào	Node (cell body): Nút Nơron t
Dendrites: Sợi nhánh	Input: Đầu vào
Synapse: Khớp thần kinh	Weights: Trọng số
Axon: Sợi trực	Output: Đầu ra

Cấu tạo nơron nhân tạo

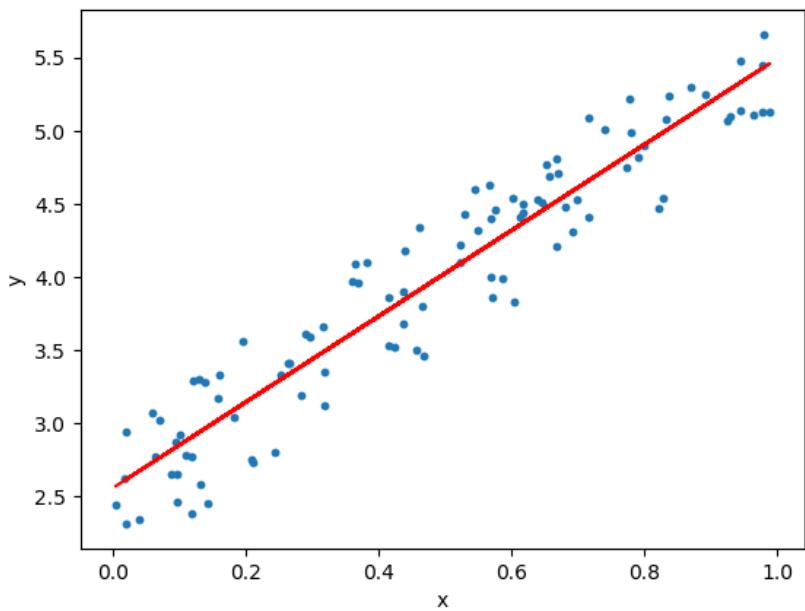


- Các giá trị x_1, x_2, \dots, x_n là các tín hiệu đầu vào
- Các giá trị w_1, w_2, \dots, w_n là các trọng số tương ứng với x_1, x_2, \dots, x_n và b là bias có thể nhận các giá trị khác 0
- $f(w, x)$ là hàm tích hợp các tín hiệu đầu vào
- Activation function được dùng để tính giá trị đầu ra của nơron dựa trên một giá trị ngưỡng (sigmoid, tanh, RELU, ...)

Cấu tạo nơron nhân tạo

- f(w,x) thường được tính toán bởi một hàm tuyến tính

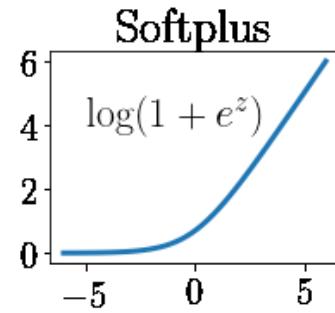
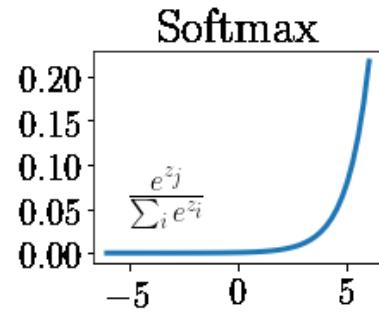
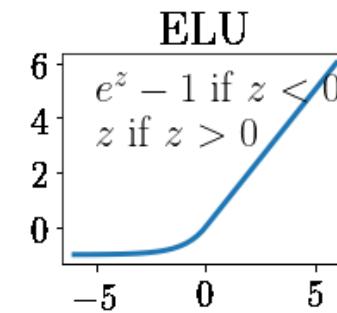
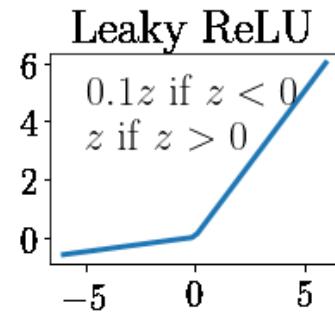
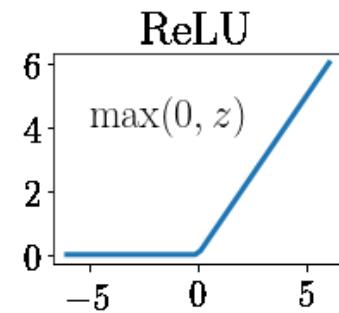
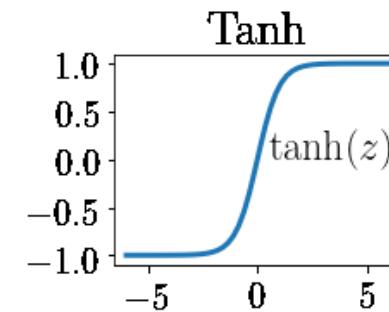
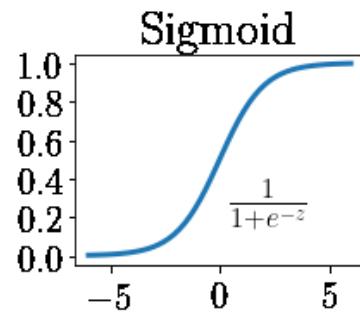
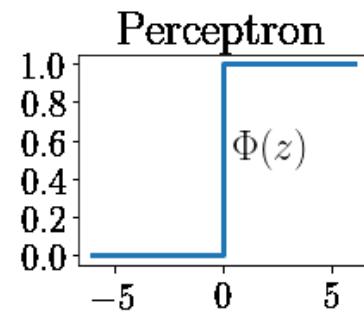
$$\begin{aligned}f(w, x) &= w_0 + \sum_{i=1}^m w_i x_i + \dots + w_m x_m \\&= w_0 1 + \sum_{i=1}^m w_i x_i = \sum_{i=0}^m w_i x_i\end{aligned}$$



$$f(w, x) = w_0 + w_1 x_1$$

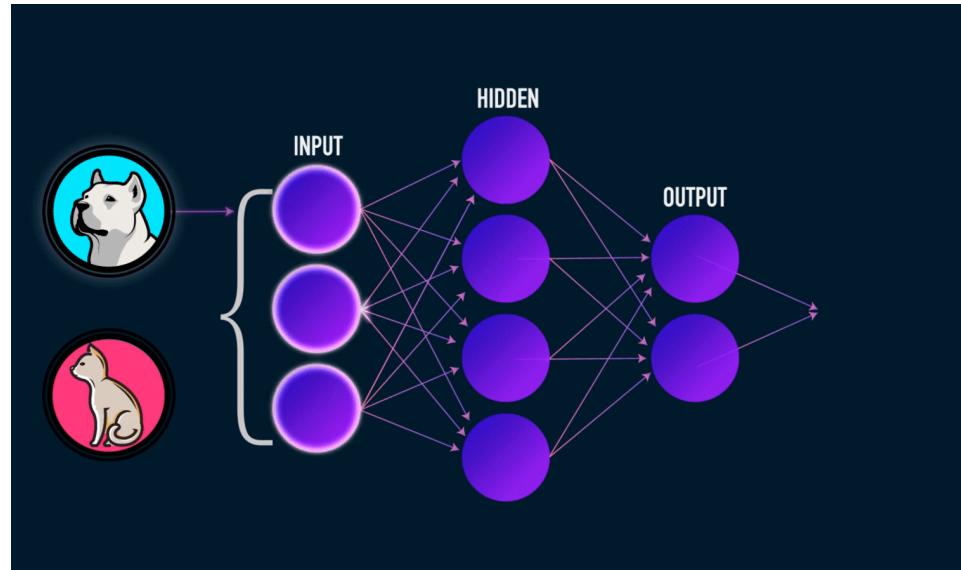
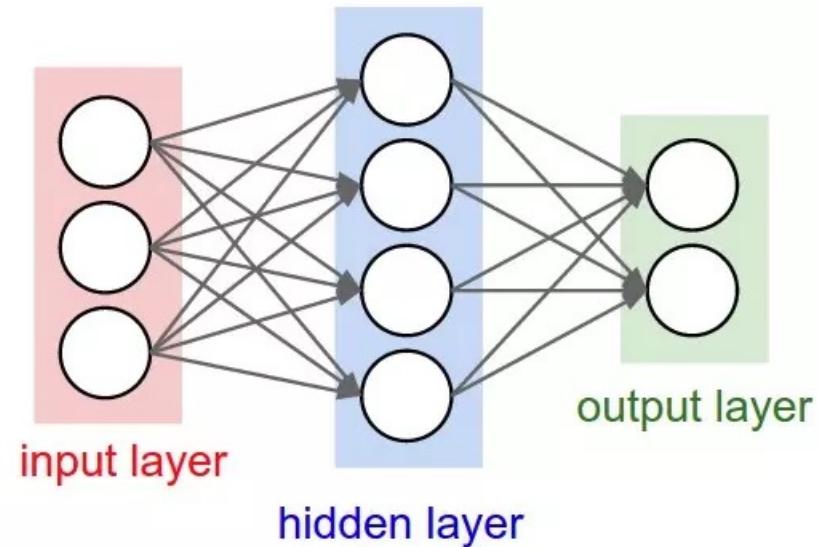
Cấu tạo mạng nơron nhân tạo

■ Các hàm activation

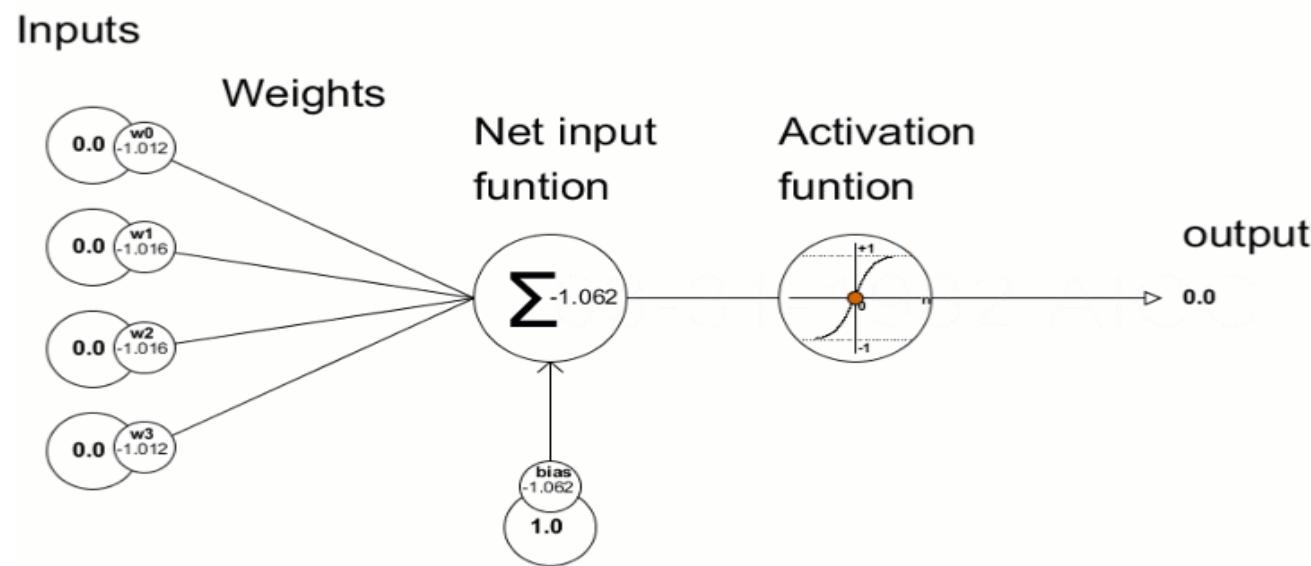


Kiến trúc mạng nơ-ron

- Một tầng (layer) chứa một nhóm các nơ-ron
- Tầng ẩn (hidden layer) là một tầng nằm ở giữa tầng đầu vào (input layer) và tầng đầu ra (output layer)
- Các nút ở tầng ẩn (hidden nodes) không tương tác trực tiếp với môi trường bên ngoài (của mạng nơ-ron)
- Một ANN được gọi là liên kết đầy đủ (fully connected) nếu mọi đầu ra từ một tầng liên kết với mọi nơ-ron của tầng kế tiếp



Kiến trúc mạng nơron

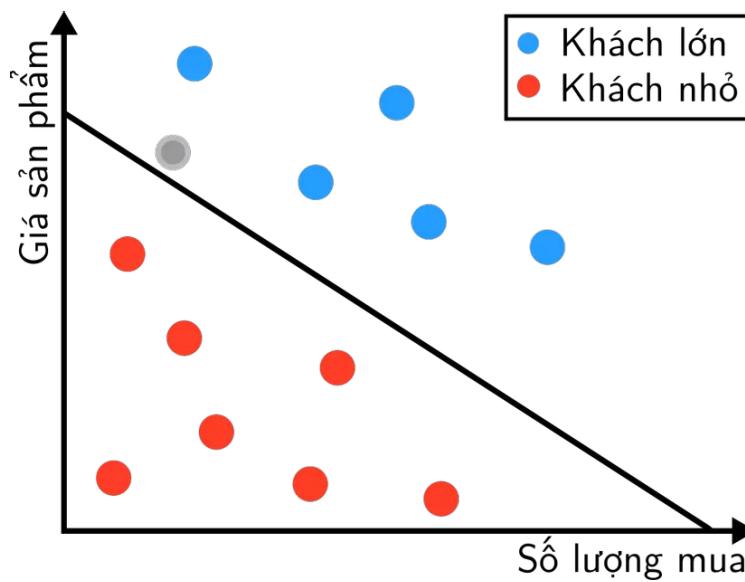


- Cách thức hoạt động
 - Mạng neural network sẽ lấy đầu ra của các layer phía trước sau đó nhân với các trọng số (weight) và tính tổng rồi cộng thêm bias
 - cuối cùng là đưa qua activation function để tạo ra đầu ra cho các layer hiện tại và đầu ra của một layer hiện tại sẽ tiếp tục làm đầu vào cho layer tiếp theo, như hình gif minh họa phía dưới.
 - Từ đó giúp mạng có thể học được những biến diễn phức tạp của data.
 - Công thức tính ouput của một layer khi nhận input từ layer phía trước là:

$$A^l = activation(W^l A^{l-1} + B^l)$$

Thuật toán PLA (Perceptron Learning Algorithm)

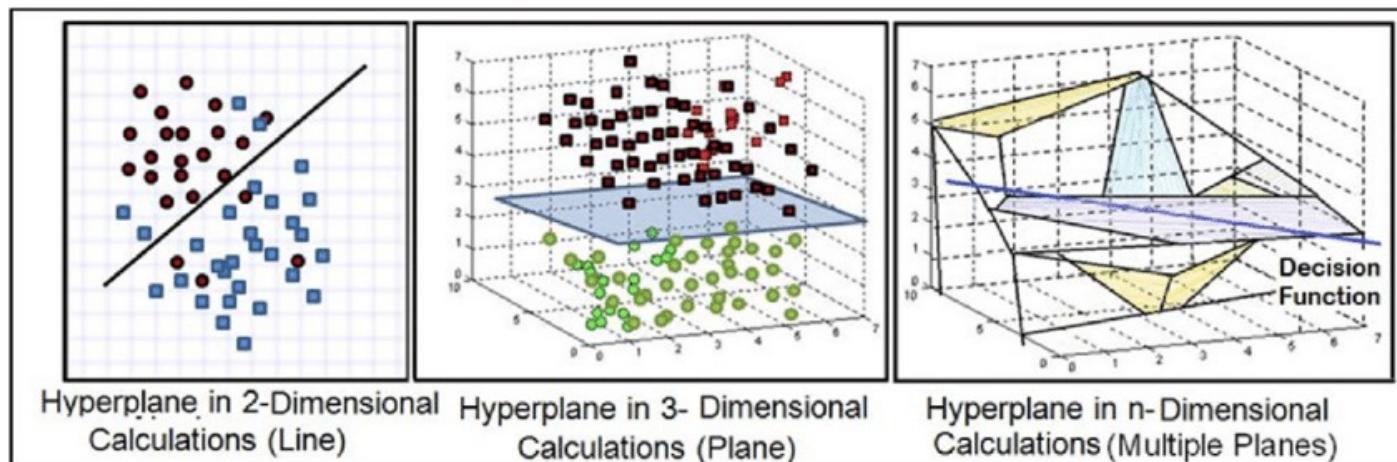
- Xét ví dụ:
- Giả sử chúng ta cần chia khách hàng ra làm hai loại/lớp (category/class) dựa vào nguồn lợi họ đem lại cho công ty: khách hàng nhỏ và khách hàng lớn. Về cơ bản, nguồn lợi được tính theo giá mặt hàng và số lượng khách mua. Như vậy, ta sẽ biểu diễn khách hàng theo hai yếu tố trên trên mặt phẳng:



Thuật toán PLA (Perceptron Learning Algorithm)

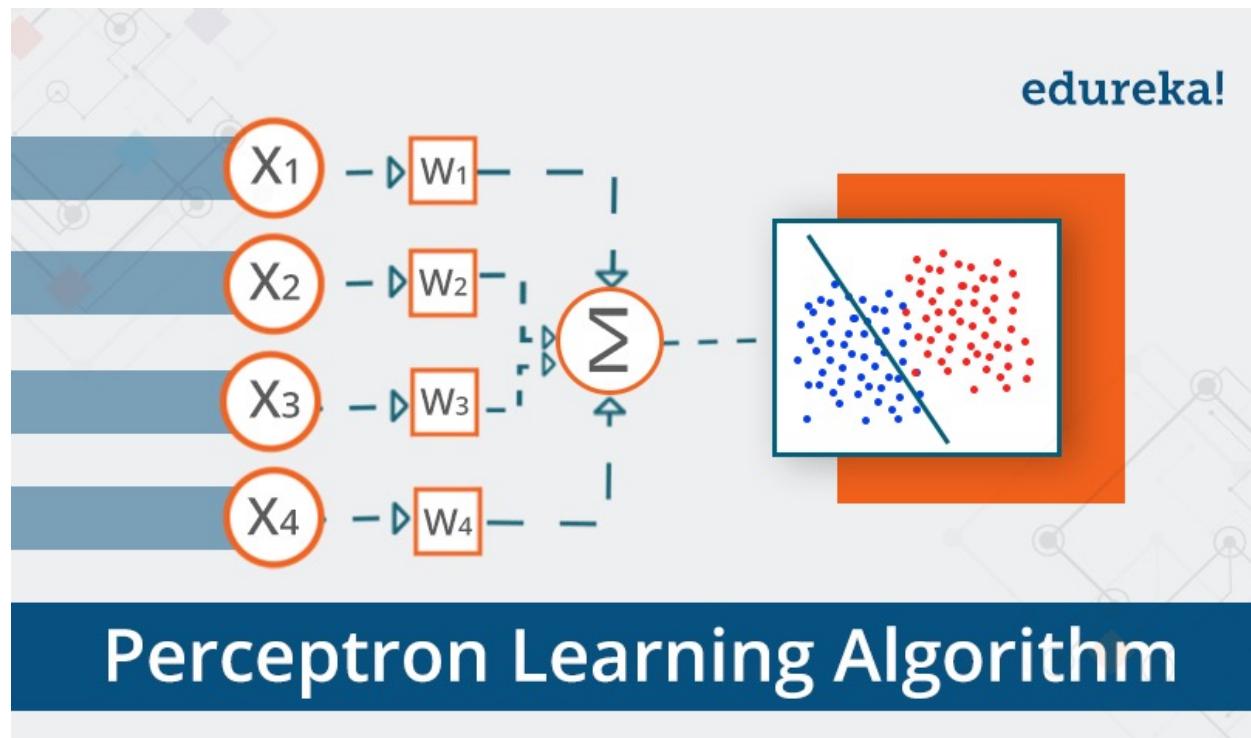
- Một số nhận xét:

- Bài toán của chúng ta là từ **những điểm xanh** và **đỏ** cho trước (tức marketer đã xác định), **hãy xây dựng một quy tắc phân loại để dự đoán class của điểm màu xám**.
- Nói cách khác, chúng ta cần xác định một **biên giới** để **chia lãnh thổ** của hai class này, rồi với điểm cần phân loại màu xám ta chỉ cần xem nó nằm ở phía bên nào của đường biên giới là xong.
- Biên giới đơn giản nhất (theo đúng nghĩa toán học)
 - ✓ trong mặt phẳng là **một đường thẳng** (đường màu đen trong hình),
 - ✓ trong không gian ba chiều là **một mặt phẳng**,
 - ✓ trong không gian nhiều chiều là **một siêu phẳng** (hyperplane, một đường thẳng nằm trong nhiều chiều).

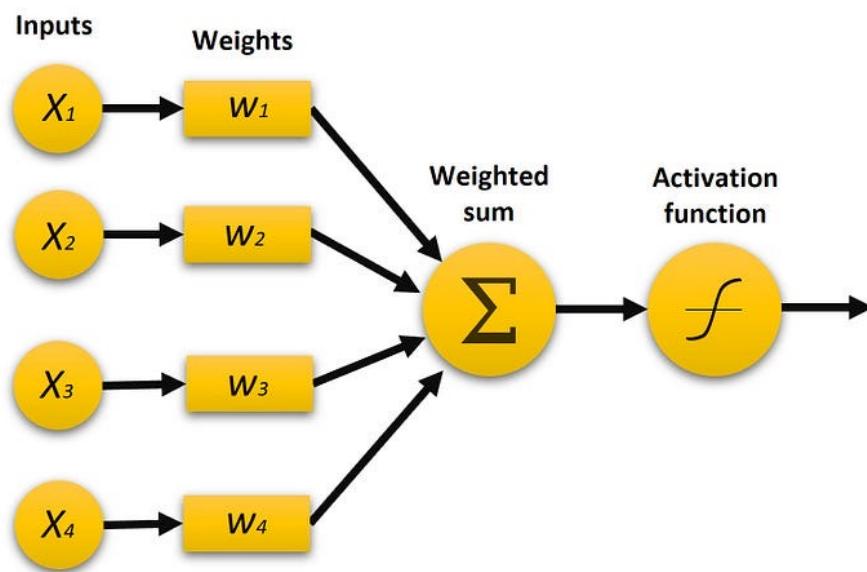


Thuật toán PLA (Perceptron Learning Algorithm)

- Thuật toán Perceptron tìm hiểu cách phân chia dữ liệu thành hai lớp bằng cách tìm một “siêu phẳng” (hyperplane) hợp lý. Trong không gian hai chiều, siêu phẳng này là một đường thẳng.



Thuật toán PLA (Perceptron Learning Algorithm)



- Một Perceptron có thể được xem như là một neuron nhân tạo đơn lẻ. Nó bao gồm:
 - Đầu vào (inputs): Mỗi đầu vào được đánh trọng số (weights) riêng biệt.
 - Tổng trọng số (weighted sum): Tính tổng các đầu vào nhân với trọng số tương ứng.
 - Hàm kích hoạt (activation function): Đơn vị tính toán sẽ quyết định neuron có nên được kích hoạt hay không.

Thuật toán PLA (Perceptron Learning Algorithm)

- Quá trình học diễn ra qua các bước sau:
 - Bước 1: Khởi tạo **trọng số** với giá trị **nhỏ** hoặc **bằng không**.
 - Bước 2: Với mỗi mẫu trong tập dữ liệu huấn luyện:
 - ✓ **Tính tổng trọng số.**

$$z = \sum_{i=1}^N w_i * x_i + b$$

- ✓ **Áp dụng hàm kích hoạt để nhận được dự đoán.**

$$\hat{y} = g(z) = \begin{cases} 1 & \text{if } z \geq 0.5 \\ 0 & \text{if } z < 0.5 \end{cases}$$

Thuật toán PLA (Perceptron Learning Algorithm)

- ✓ Cập nhật trọng số dựa trên **sự chênh lệch** giữa **dự đoán** và **nhãn thực tế**.

- **Tính hàm lỗi**

$$L = \frac{1}{2N} \sum (y - \hat{y})^2$$

- **Thực hiện chain rule**

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_i} = \frac{1}{N} (\hat{y} - y_i) x_i$$

- **Công thức cập nhật trọng số**

$$w_{new} = w_{old} + \eta * (y - \hat{y}) * x$$

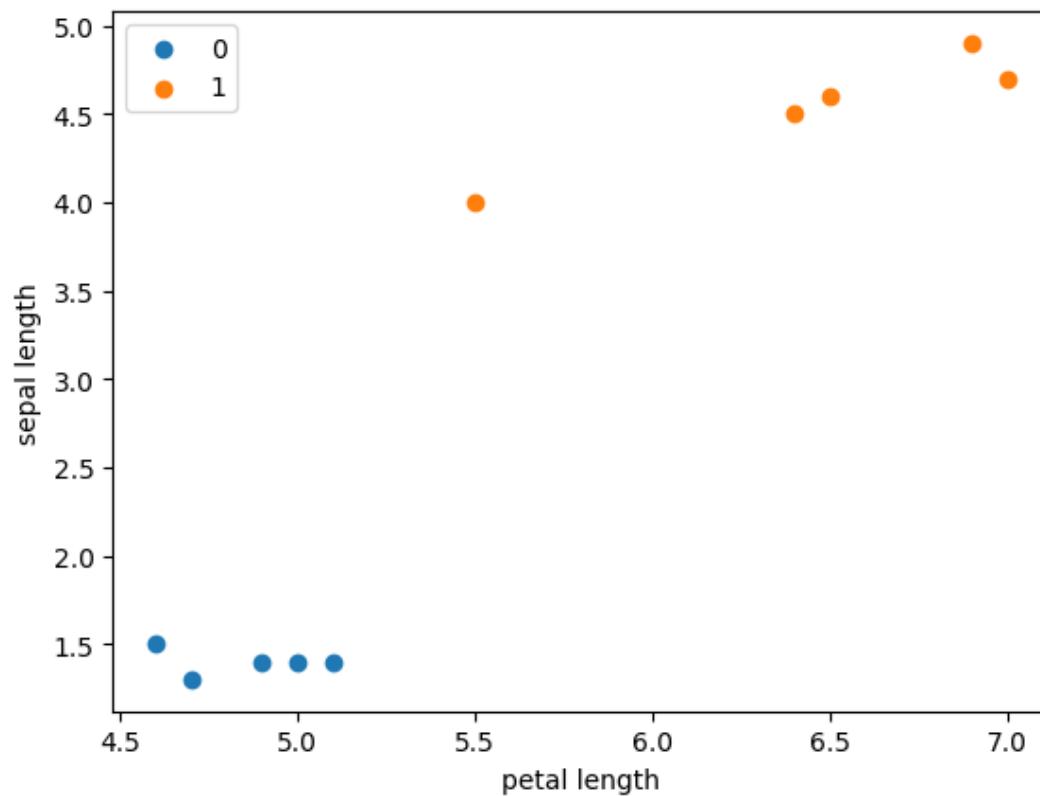
$$b_{new} = b_{old} + \eta * (y - \hat{y}) * x$$

- Lặp lại bước 2 cho đến khi **đạt được số lượt lặp tối đa** hoặc **lỗi trên tập huấn luyện không thay đổi**.

Thuật toán PLA (Perceptron Learning Algorithm)

■ Tập dữ liệu

X1	X2	Y
5.1	1.4	0.
4.9	1.4	0.
4.7	1.3	0.
4.6	1.5	0.
5.	1.4	0.
7.	4.7	1.
6.4	4.5	1.
6.9	4.9	1.
5.5	4.	1.
6.5	4.6	1.

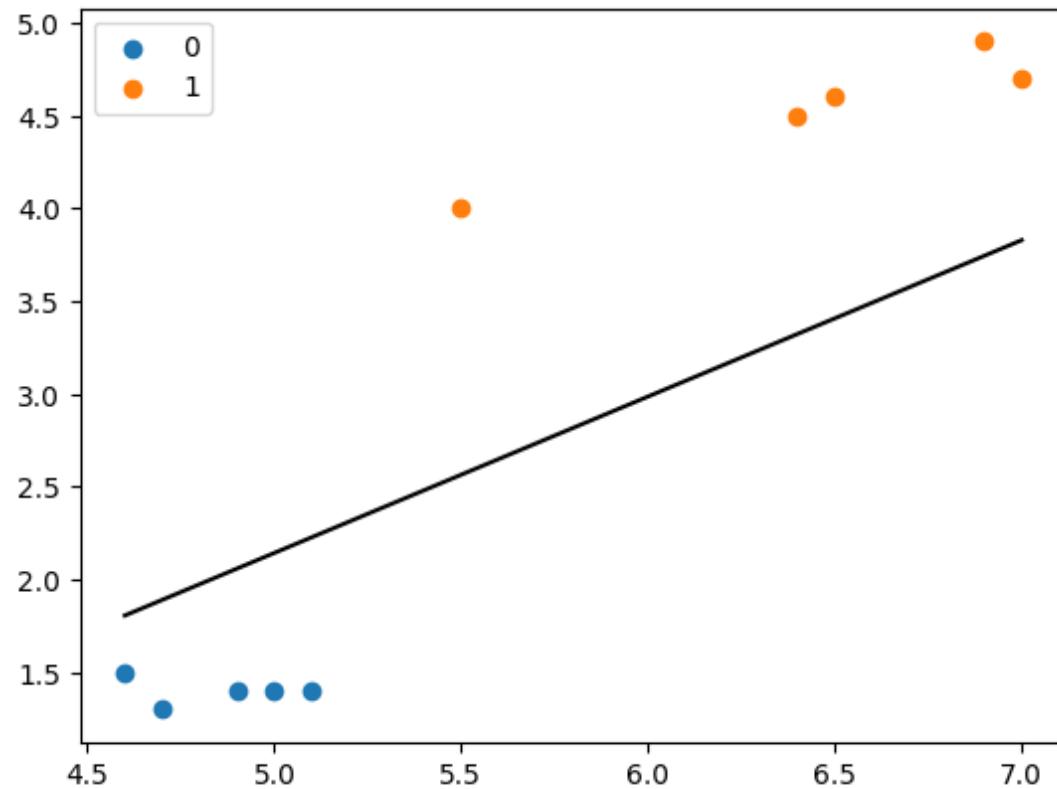


$X = [4.6, 1.4] \Rightarrow Y = 0 \text{ or } 1 ?$

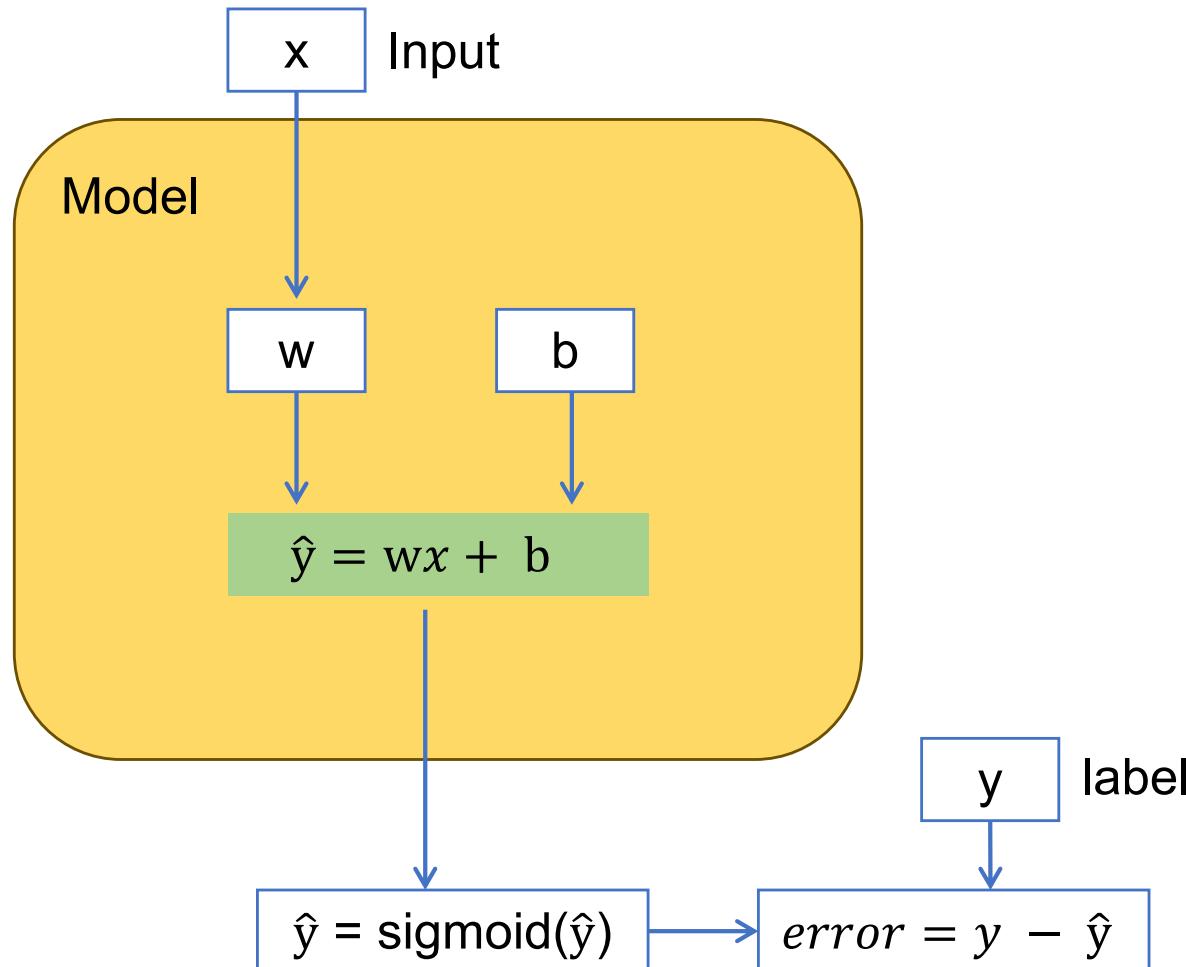
Thuật toán PLA (Perceptron Learning Algorithm)

■ Tập dữ liệu

X1	X2	Y
5.1	1.4	0.
4.9	1.4	0.
4.7	1.3	0.
4.6	1.5	0.
5.	1.4	0.
7.	4.7	1.
6.4	4.5	1.
6.9	4.9	1.
5.5	4.	1.
6.5	4.6	1.



Thuật toán PLA (Perceptron Learning Algorithm)



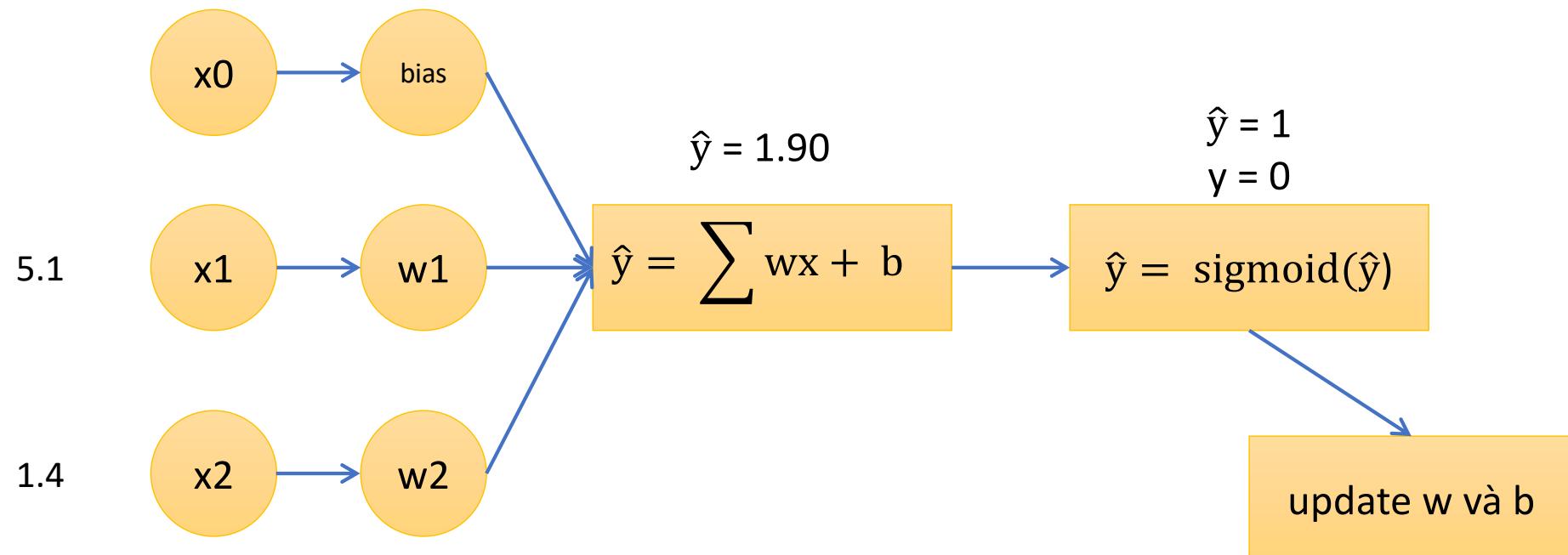
- 1) Pick a sample (x,y) from training data
- 2) Compute the output \hat{y}
$$\hat{y} = wx + b$$
- 3) Activation
$$\hat{y} = \text{sigmoid}(\hat{y})$$
- 4) Compute error:
$$error = y - \hat{y}$$
- 5) Update parameters
$$w_{\text{new}} = w_{\text{old}} + \eta * (y - \hat{y}) * x$$

$$b_{\text{new}} = b_{\text{old}} + \eta * (y - \hat{y}) * x$$

Thuật toán PLA (Perceptron Learning Algorithm)

Bước 1: lr = 0.01, epoch=20

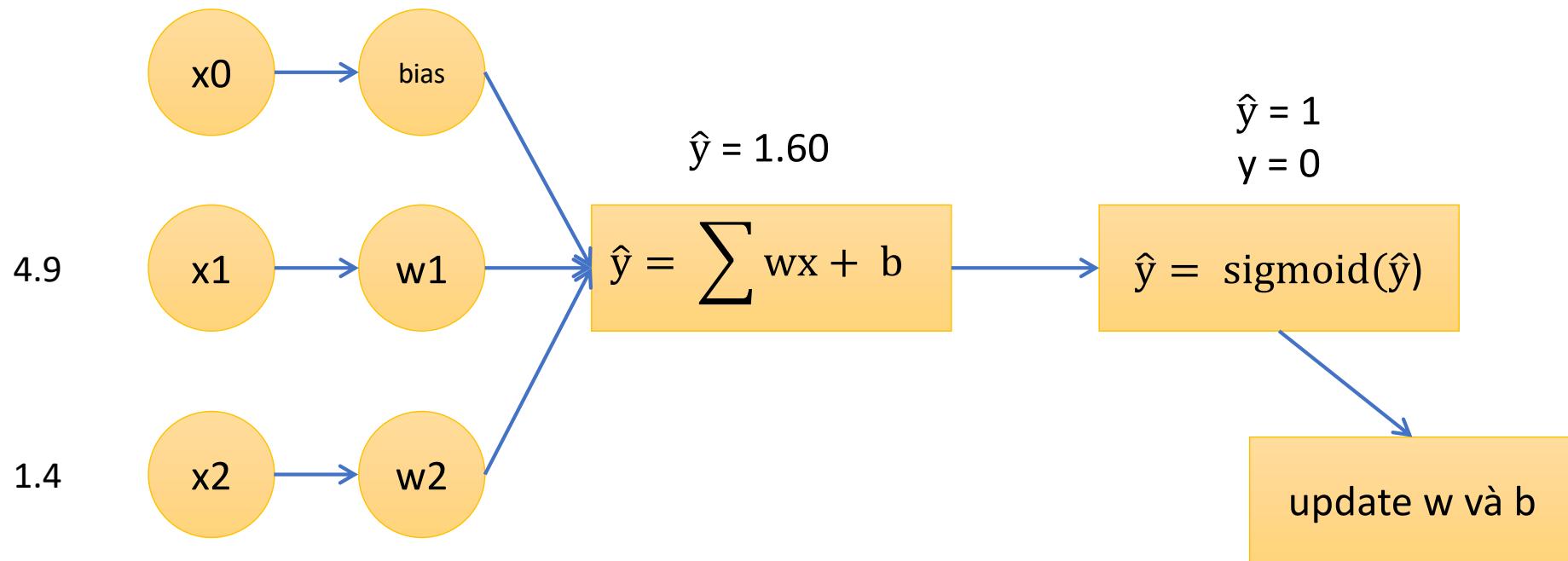
$$w = [0.94 \ -0.88], \text{bias}:-0.98$$



$$w = [0.12, 0.28], \text{bias} = 0.61$$

Thuật toán PLA (Perceptron Learning Algorithm)

$w = [0.12 \ 0.28]$, bias = 0.61



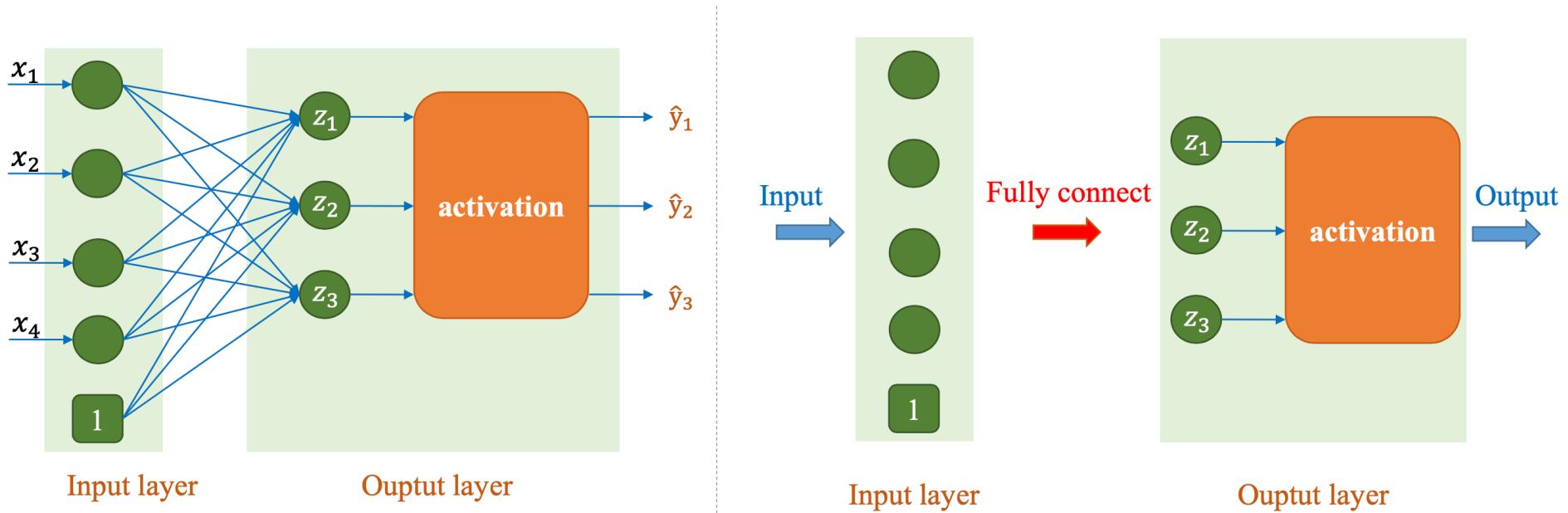
$w = [0.07, 0.27]$, bias = 0.59

Thuật toán PLA (Perceptron Learning Algorithm)

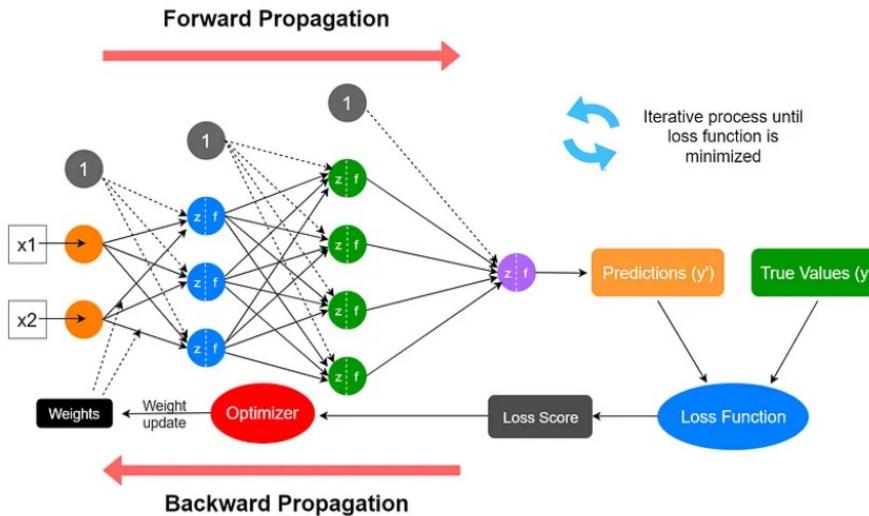
- Cài đặt python

Thuật toán MLPs(mạng nơron đa tầng)

❖ Softmax regression



Thuật toán MLPs(mạng nơron đa tầng)



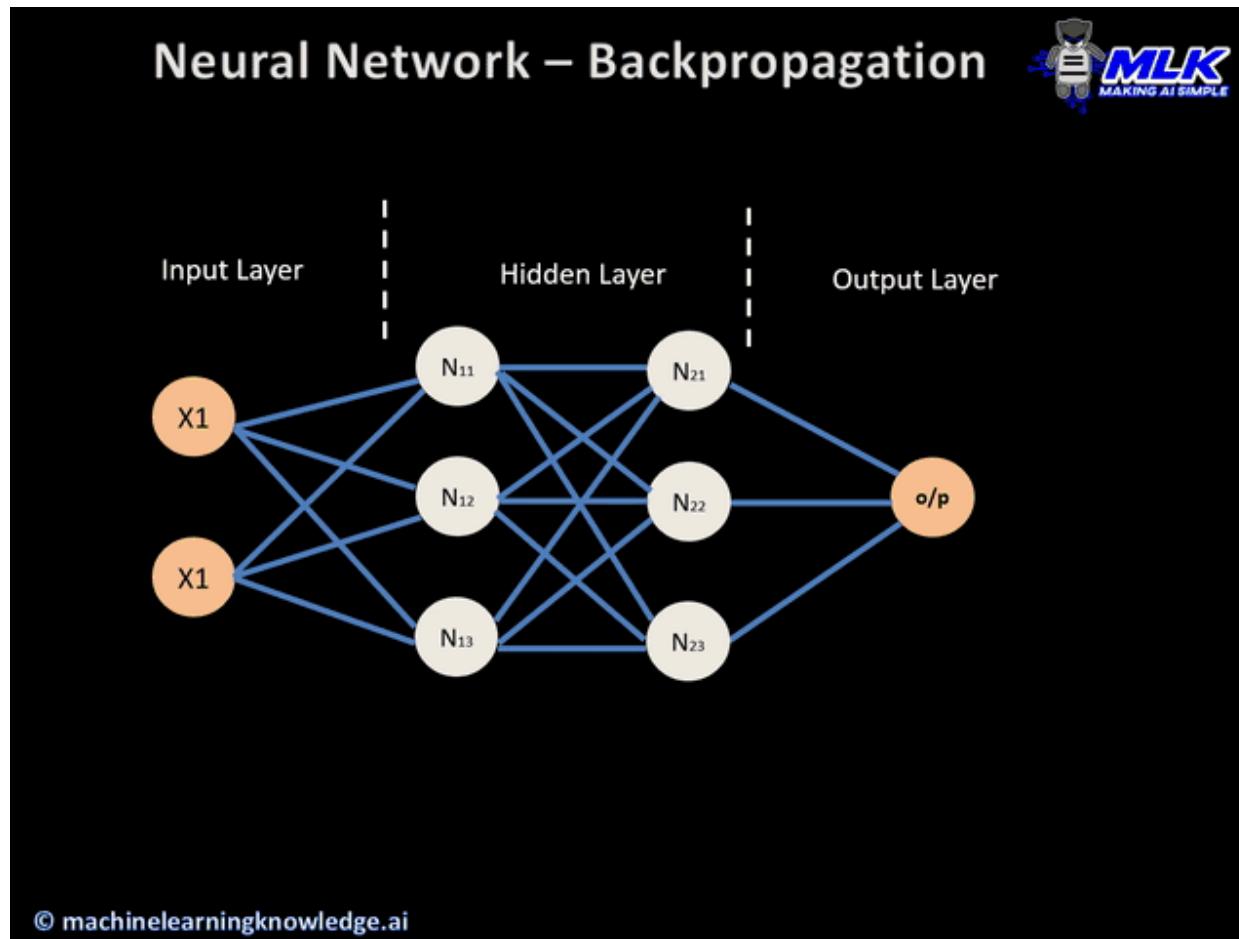
- Mạng neural network đơn giản được tạo ra từ các liên kết giữa các perceptron. Mạng neural network này có tên là Multi-layer Perceptron (MLP)
- Gọi là Multi-layer Perceptron (perceptron nhiều lớp) bởi vì nó là tập hợp của các perceptron chia làm nhiều nhóm, mỗi nhóm tương ứng với một layer.
- Trong hình trên ta có một ANN với 3 lớp: Input layer (lớp đầu vào), Output layer (lớp đầu ra) và Hidden layer (lớp ẩn).
- Thông thường khi giải quyết một bài toán ta chỉ quan tâm đến input và output của một model, do vậy trong MLP nói riêng và ANN nói chung ngoài lớp Input và Output ra thì các lớp neuron ở giữa được gọi chung là Hidden (ẩn không phải là không nhìn thấy mà đơn giản là không quan tâm đến).

Thuật toán MLPs(mạng nơ-ron đa tầng)

- Một mạng nơ-ron nhiều tầng (multi-layer NN) được học bởi giải thuật lan truyền ngược (Back Propagation - BP) có thể biểu diễn một hàm phân tách phi tuyến phức tạp (highly non-linear separation function)
- Giải thuật học BP được sử dụng để học các trọng số của một mạng nơ-ron nhiều tầng
 - Cấu trúc mạng cố định (các nơ-ron và các liên kết giữa chúng là cố định)
 - Đối với mỗi nơ-ron, hàm tác động phải có đạo hàm liên tục
- Giải thuật BP áp dụng chiến lược gradient descent trong quy tắc cập nhật các trọng số
 - Để cực tiểu hóa lỗi (khác biệt) giữa các giá trị đầu ra thực tế và các giá trị đầu ra mong muốn, đối với các ví dụ học

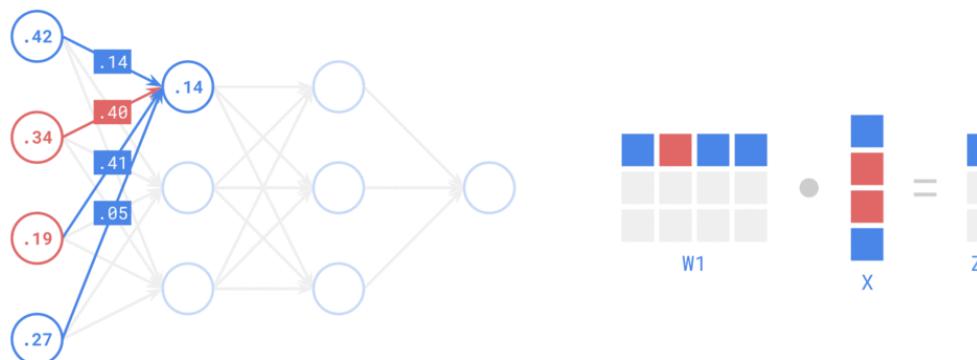
Thuật toán MLPs(mạng nơron đa tầng)

- Giải thuật học lan truyền ngược tìm kiếm một vectơ các trọng số (weights vector) giúp cực tiểu hóa lỗi tổng thể của hệ thống đối với tập học



Thuật toán MLPs(mạng nơron đa tầng)

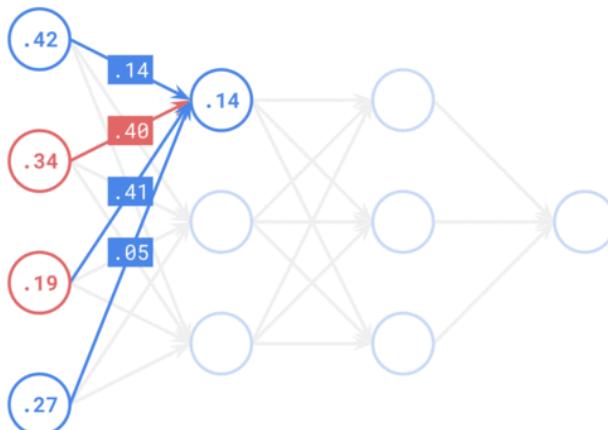
- Giải thuật BP bao gồm 2 giai đoạn (bước)
 - Giai đoạn lan truyền tín hiệu (Feedforward). Các tín hiệu đầu vào (vectơ các giá trị đầu vào) được lan truyền tiến từ tầng đầu vào đến tầng đầu ra (đi qua các tầng ẩn)
 - Trong toàn bộ các nốt mạng nơron đều có thể kết hợp đôi một với nhau theo một chiều duy nhất từ tầng vào đến tầng ra. Có nghĩa là, mỗi node ở một tầng sẽ nhận đầu vào là tất cả các nốt ở tầng trước đó và ngược lại. Có nghĩa là, việc suy luận Neural Network là dạng suy luận tiến (feedforward).



Thuật toán MLPs(mạng nơron đa tầng)

- Thuật toán FeedForward

- Ý tưởng: Trong toàn bộ các nốt mạng nơ ron đều có thể kết hợp đôi một với nhau theo một chiều duy nhất từ tầng vào đến tầng ra. Có nghĩa là, mỗi node ở một tầng sẽ nhận đầu vào là tất cả các nốt ở tầng trước đó và ngược lại. Có nghĩa là, việc suy luận Neural Network là dạng suy luận tiến (feedforward).

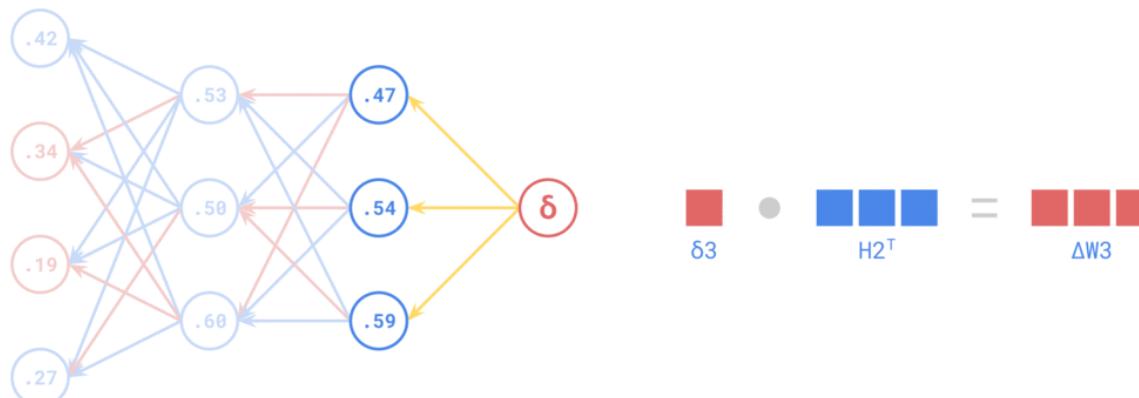


$$\begin{matrix} \text{W1} & \cdot & \text{X} & = & \text{Z1} \end{matrix}$$

The diagram shows a matrix multiplication operation. On the left is a weight matrix W_1 with four columns. To its right is a dot symbol indicating multiplication. Next is the input vector X , which is a vertical column with three red squares at the top and one blue square at the bottom. An equals sign follows, leading to the result Z_1 , which is a vertical column with one blue square at the top and two grey squares at the bottom.

Thuật toán MLPs(mạng nơron đa tầng)

- Giai đoạn lan truyền ngược (Backpropagation)
 - ✓ Căn cứ vào giá trị đầu ra mong muốn của vectơ đầu vào, hệ thống tính toán giá trị lỗi
 - ✓ Bắt đầu từ tầng đầu ra, giá trị lỗi được lan truyền ngược qua mạng, từ tầng này qua tầng khác (phía trước), cho đến tầng đầu vào
 - ✓ Việc lan truyền ngược lỗi (error back-propagation) được thực hiện thông qua việc tính toán (một cách truy hồi) giá trị gradient cục bộ của mỗi nơ-ron



Thuật toán MLPs(mạng nơron đa tầng)

- Công thức hàm tính tổng trọng số

$$z = \sum w * x + b$$

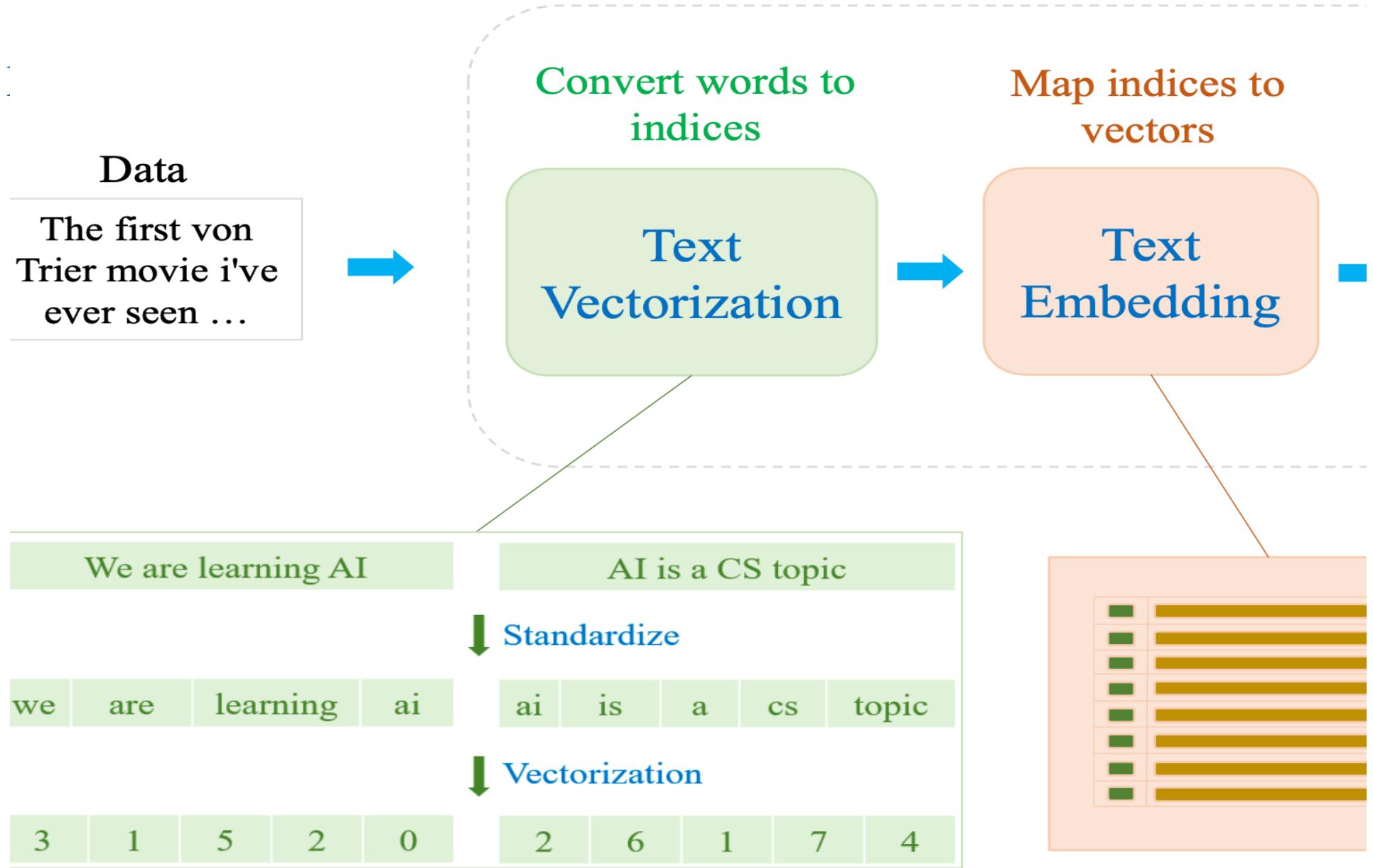
- Công thức hàm loss

$$L = \frac{1}{2N} \sum (y - \hat{y})^2$$

- Công thức hàm activate

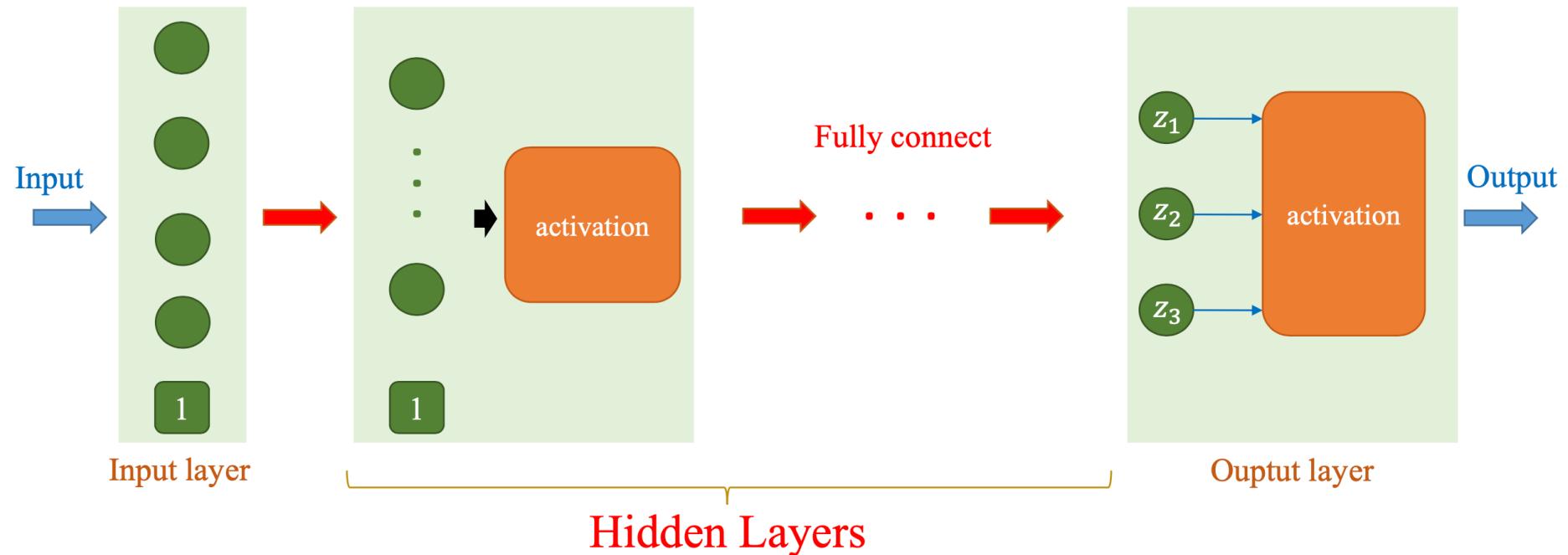
$$\hat{y} = g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Thuật toán MLPs(mạng nơron đa tầng)



Thuật toán MLPs(mạng nơron đa tầng)

Model (Network) Construction



How many hidden layers?
How many nodes in a hidden layer?

Which activation function?
Which network components?

Year 2020

Thuật toán MLPs(mạng nơron đa tầng)

❖ ReLU function

$$\text{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

relu
derivative

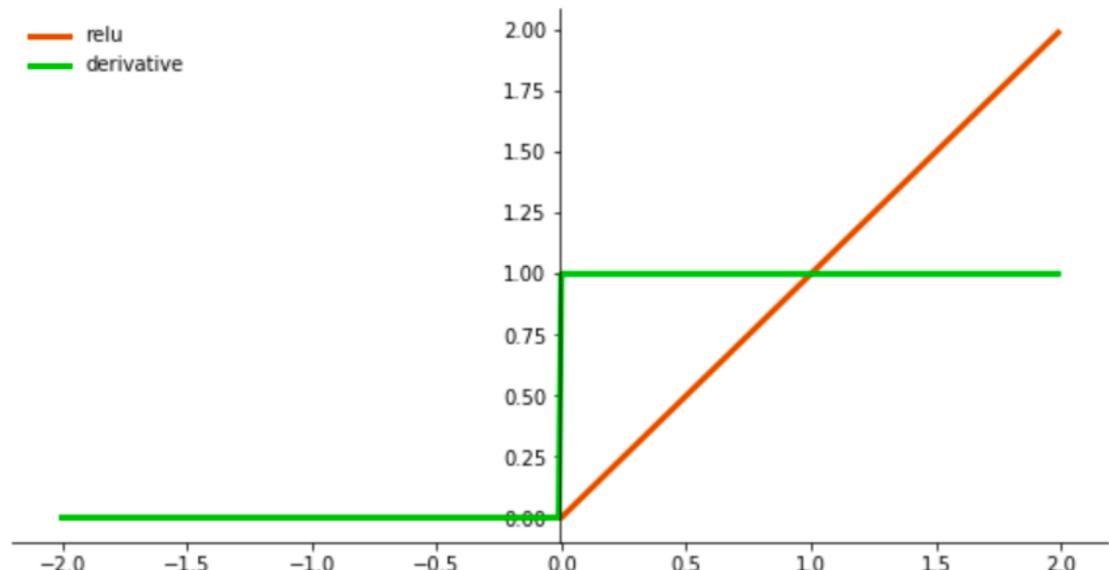
data =

1	5	-4	3	-2
---	---	----	---	----

data_a = **ReLU(data)**

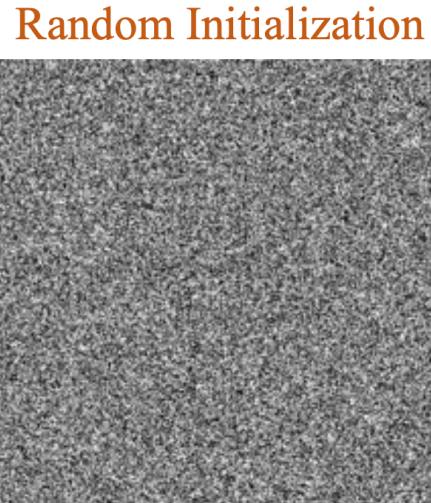
data_a =

1	5	0	3	0
---	---	---	---	---



Thuật toán MLPs(mạng nơron đa tầng)

Parameter Initialization



<https://en.wikipedia.org/wiki/Randomness>

ear 2020

initializers

Overview

deserialize

get

GlorotNormal

GlorotUniform

he_normal

he_uniform

Identity

Initializer

lecun_normal

lecun_uniform

Orthogonal

serialize

TruncatedNormal

VarianceScaling

Initialization method supported in Tensorflow

https://www.tensorflow.org/api_docs/python/tf/keras/initializers

18

Thuật toán MLPs(mạng nơron đa tầng)

Metric Selection

Compute the goodness of the current model

Useful for developers

Precision	True Positives	
	False Positives	
True Negatives	Recall	
		Accuracy
False Negatives		Root Mean Squared Error
Precision At Recall		Mean Absolute Error

www.zer0.com

Thuật toán MLPs(mạng nơron đa tầng)

- <http://playground.tensorflow.org/#activation=sigmoid®ularization=L1&batchSize=1&dataset=gauss®Dataset=reg-plane&learningRate=0.01®ularizationRate=0&noise=0&networkShape=4&seed=0.89561&showTestData=false&discretize=false&percTrainData=40&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false>

Thuật toán MLPs(mạng nơron đa tầng)

- Cài đặt python