

# Ứng dụng Trí tuệ nhân tạo trong Nuôi trồng thủy sản

NGUYỄN HẢI TRIỀU<sup>1</sup>

<sup>1</sup> Bộ môn Kỹ thuật phần mềm,  
Khoa Công nghệ thông tin, Trường ĐH Nha Trang

NhaTrang, September 2024

Trong chương này chúng ta sẽ tập trung vào **Single-layer neural networks** và mô hình phân loại **Logistic regression**.

## 1 Using Logistic Regression for Classification

- Single Layer Neural Networks
- Activation Function
- Logistic regression loss function
- Model Training with Stochastic Gradient Descent

Logistic regression, Perceptron and Linear regression are single layer neural networks.

Regression

Classification

Linear regression

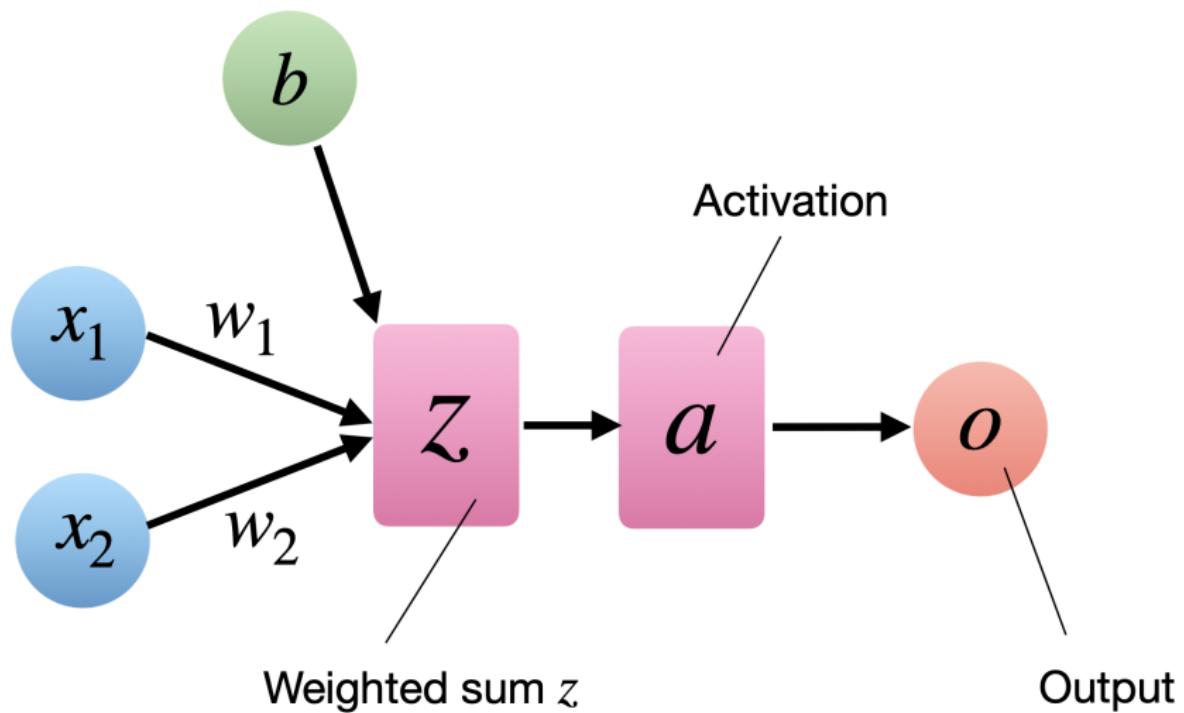
Perceptron

Logistic Regression

(a)

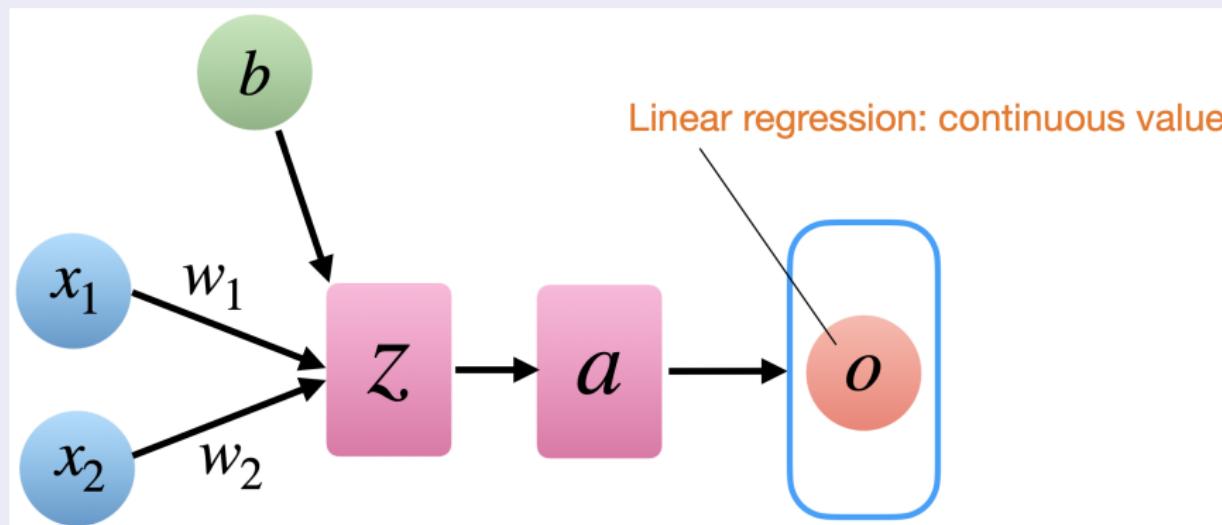
(b)

Hình 1: Các loại mô hình mạng neural 1 lớp.



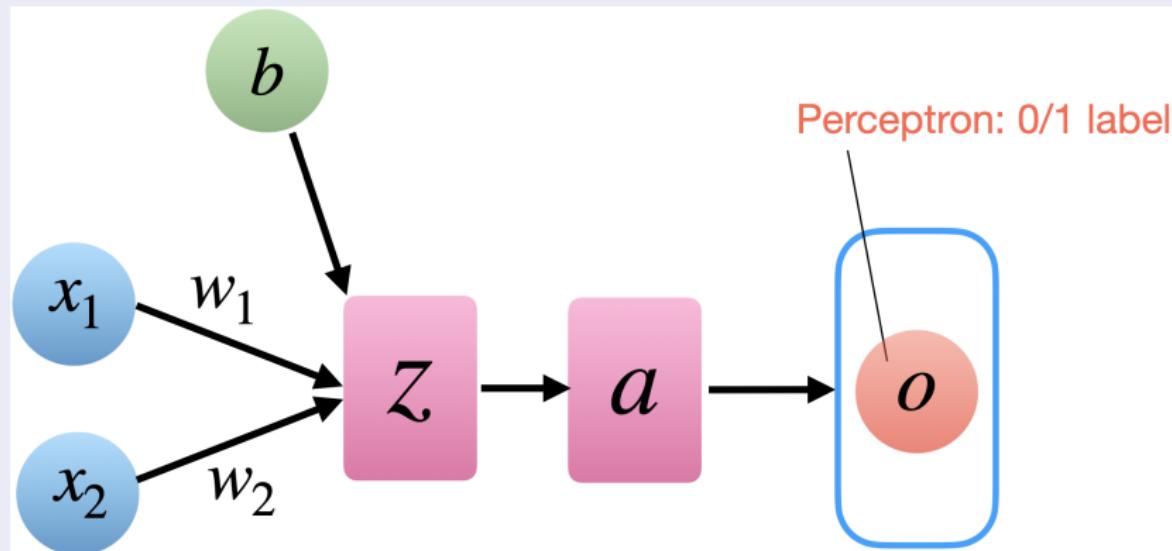
Hình 2: General Structure of Single Layer Neural Nets

**Linear regression** can be seen as a single-layer neural network where the *output is a continuous value*.



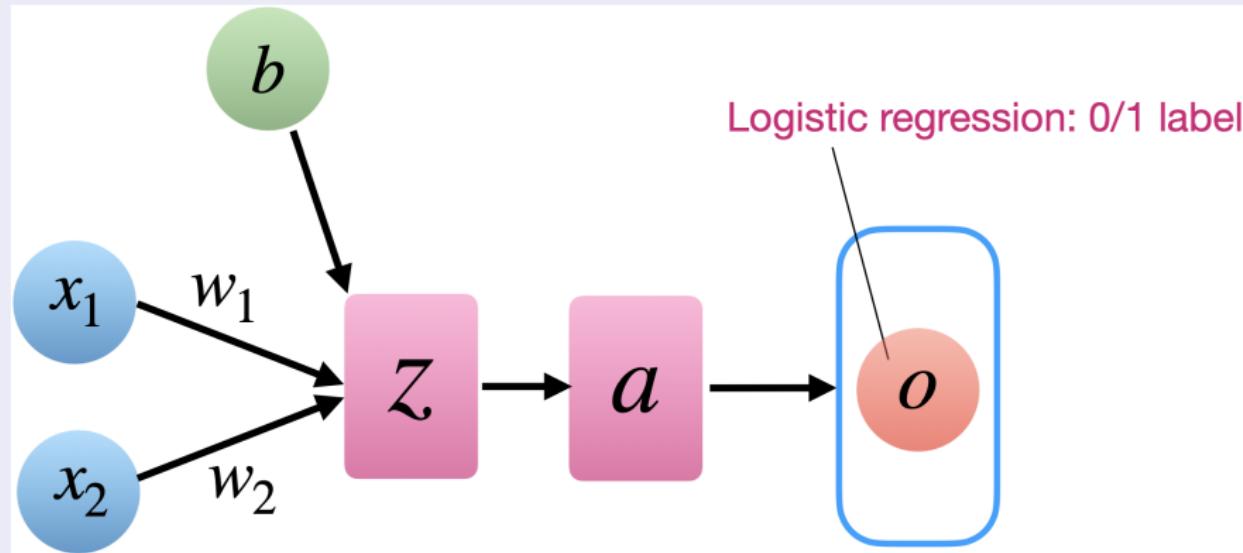
Hình 3: Linear Regression Outputs: đầu ra là các giá trị liên tục

**Perceptron** cũng là một mô hình mạng 1 lớp nhưng *đầu ra là class label (lớp 0 hoặc 1)*.



Hình 4: Perceptron Outputs

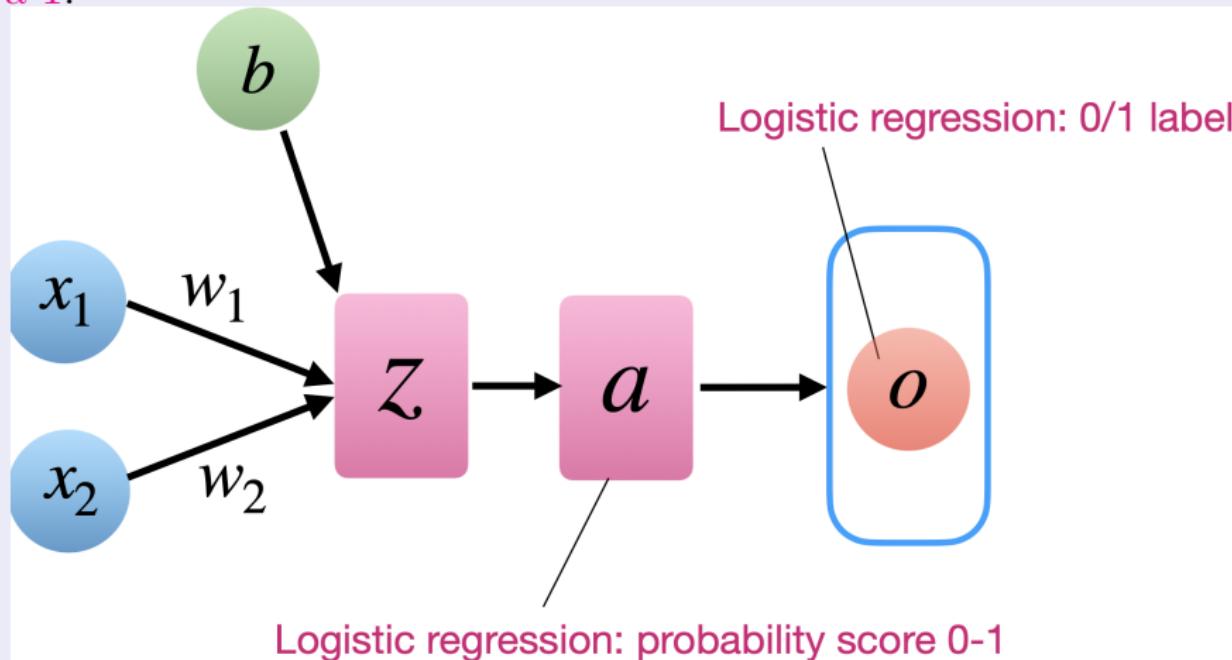
**Logistic Regression** tương tự như Perceptron là một mô hình phân lớp nhị phân, có *đầu ra là class label (lớp 0 hoặc 1)*.



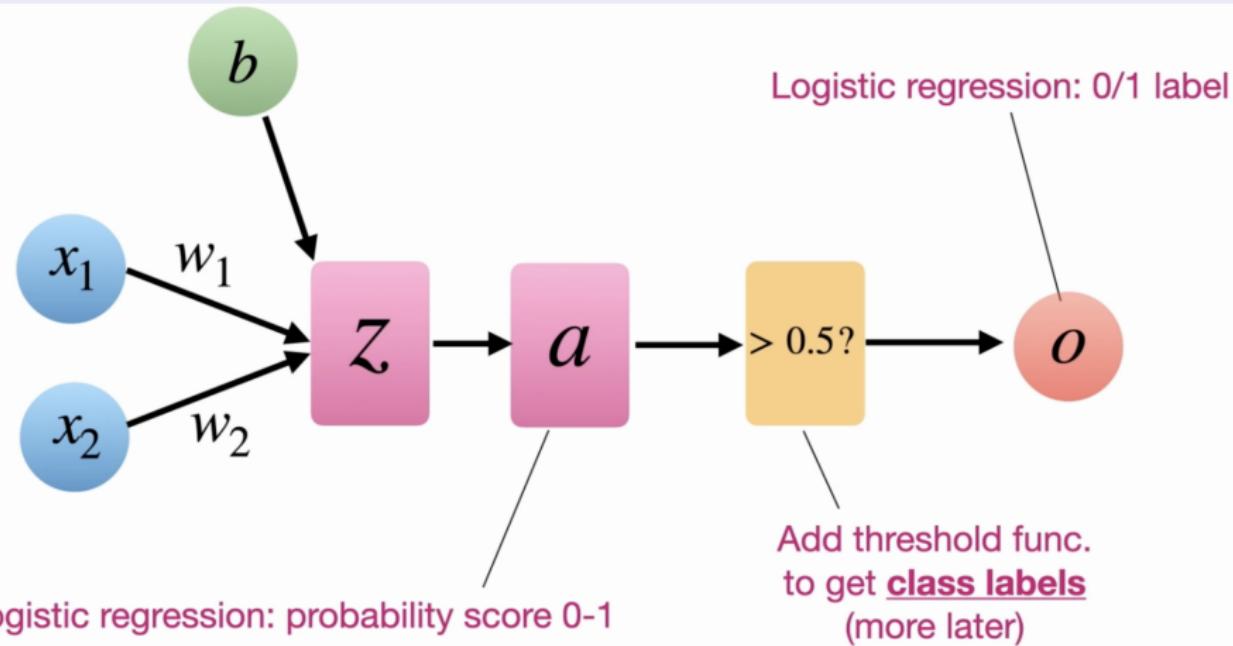
Hình 5: Logistic Regression Outputs

Lưu ý: *cần phân biệt tránh nhầm lẫn giữa Logistic và Linear regression*. Về bản chất Logistic là một mô hình phân loại.

Tuy nhiên, Logistic regression khác Perceptron ở chỗ: activation function của logistic regression là một giá trị xác suất nằm giữa 0 và 1.

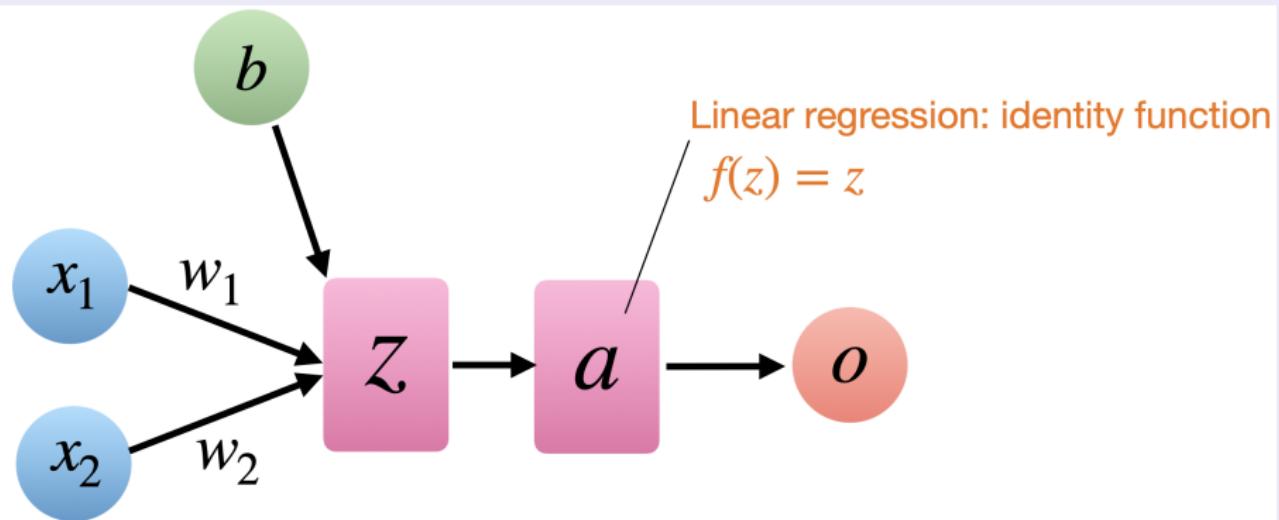


There is actually also a threshold function here, which checks whether the probability score is greater than 0.5 or not.



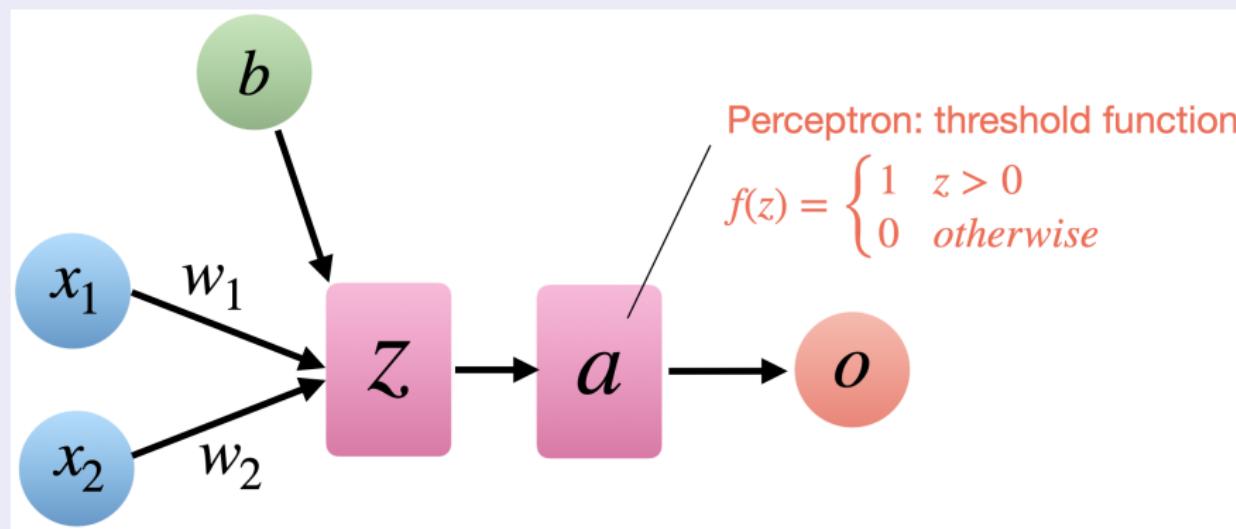
Hình 6: Cụ thể, mô hình Logistic regression cũng có threshold function như Perceptron

Trong phần này chúng ta sẽ thảo luận về **hàm kích hoạt–activation function** của các mô hình. Cụ thể, trong Linear regression, activation func. là một hàm đồng nhất  $f(z) = z$



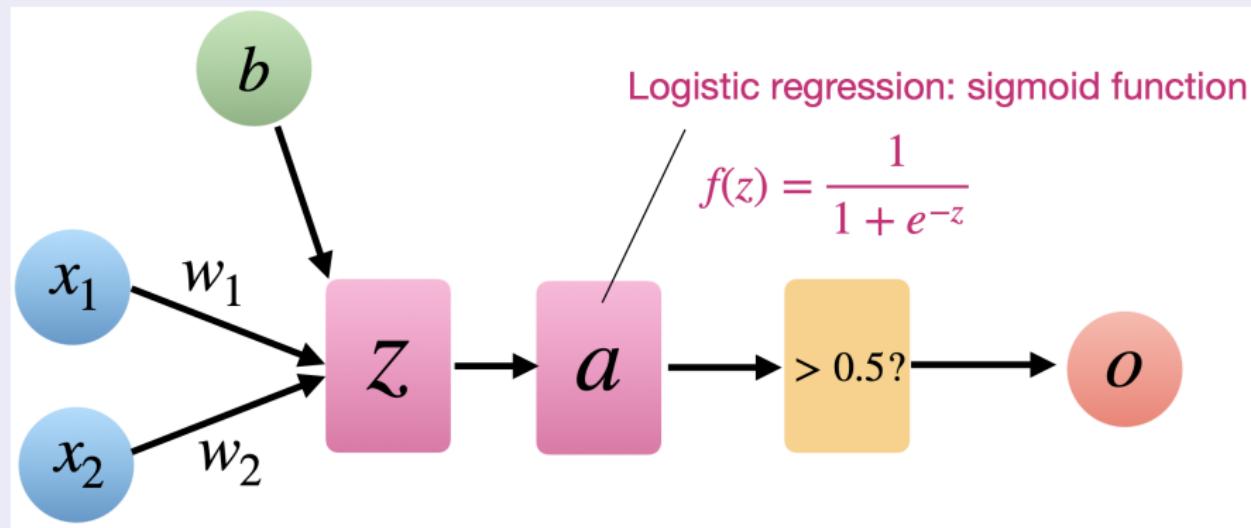
**Hình 7:** Trong thực tế chúng ta có thể bỏ qua activation function của thuật toán Linear regression, output chính là giá trị của  $z$ .

In the case of the perceptron, we have an activation function that is a threshold function.



Hình 8: Hàm kích hoạt của thuật toán Perceptron.

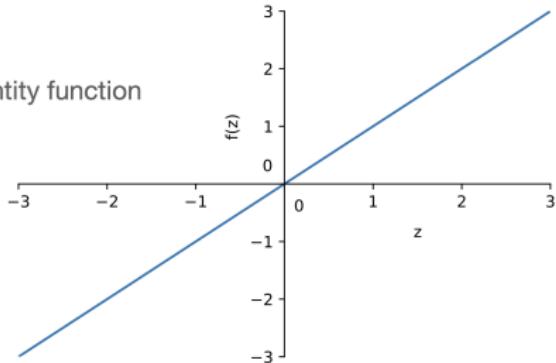
Hàm kích hoạt của thuật toán Logistic regression là hàm phi tuyến (nonlinear) thường có tên gọi là: **logistic sigmoid function**.



Hình 9: Logistic sigmoid function: this activation is followed by the threshold function  $> 0.5?$ , which checks whether the activation is greater than 0.5 or not. And based on that, the class label is then determined.

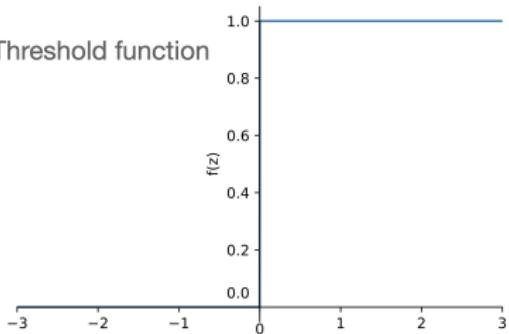
## Visualization of the activation functions

Identity function

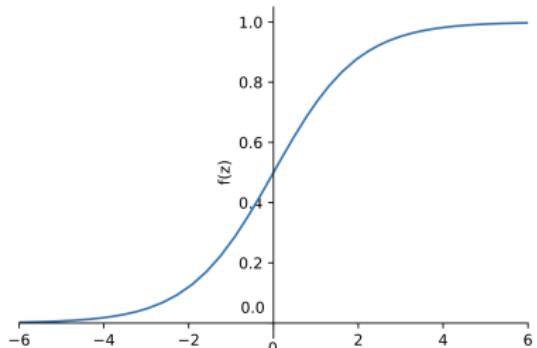


(a) Linear regression

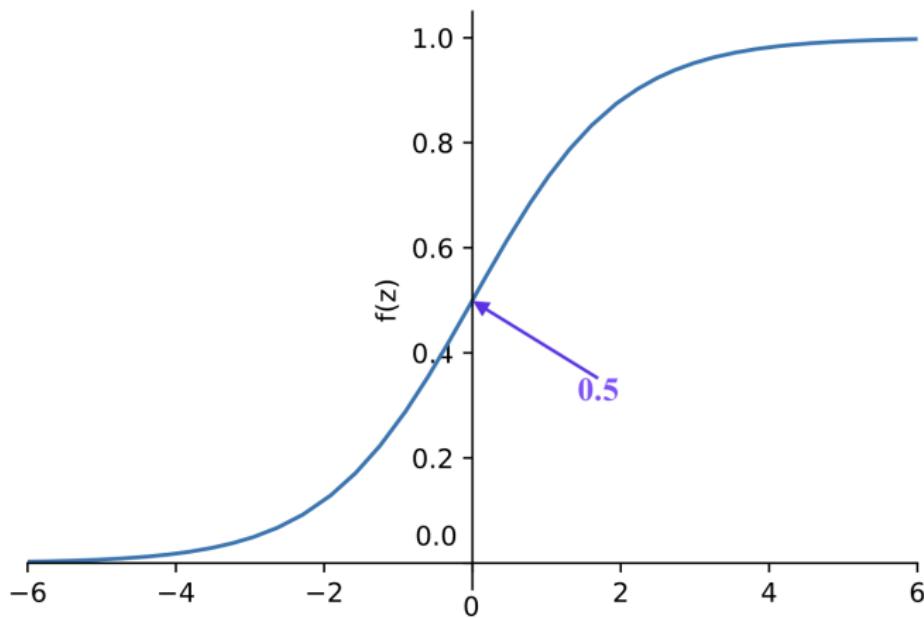
Threshold function



(b) Perceptron



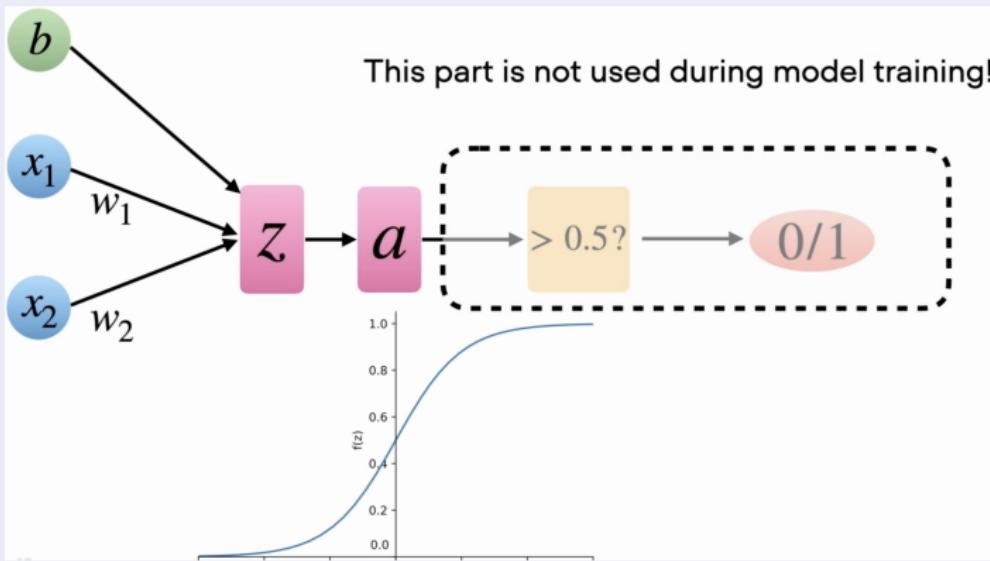
(c) Logistic regression: sigmoid



Hình 10: sigmoid func.: nếu weighted sum  $z > 0$  thì  $f(z) > 0.5$  và ngược lại. Theo quy ước kí hiệu trong ML, chúng ta thường **sử dụng**  $\sigma(z)$  thay cho  $f(z)$

# How do we actually train the logistic regression model?

Logistic regression loss function được sử dụng cho quá trình huấn luyện bài toán phân loại. Ngoài ra nó cũng được sử dụng cho việc huấn luyện các Deep neural networks.



Hình 11: We only need the weighted sum, the activation function, the true class labels for loss function.

Instead of optimizing the error or accuracy as Perceptron, which we then use to update the model weights, logistic regression uses a proxy (surrogate) loss function. And this **loss function** is related to the goal that we have, maximizing accuracy.

The loss is also known as **negative log-likelihood** and **binary cross-entropy loss**.

- negative log likelihood is a term from statistics.
- binary cross entropy was derived from information theory.

What is our overall goal when we train machine learning classifiers?

*The overall goal is to predict classes correctly or make correct predictions.* In Perceptron or Logistic regression, our goal is to predict the class labels zero and one **correctly**.

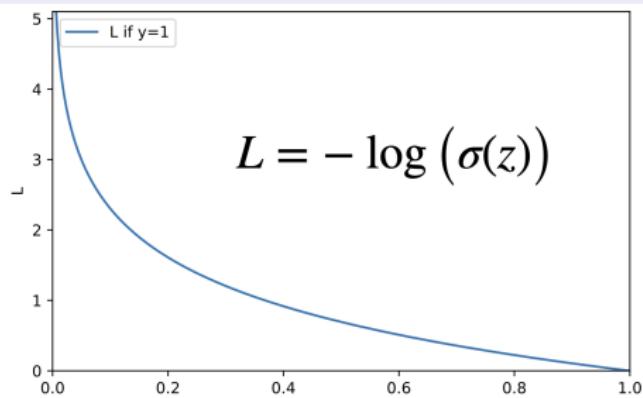
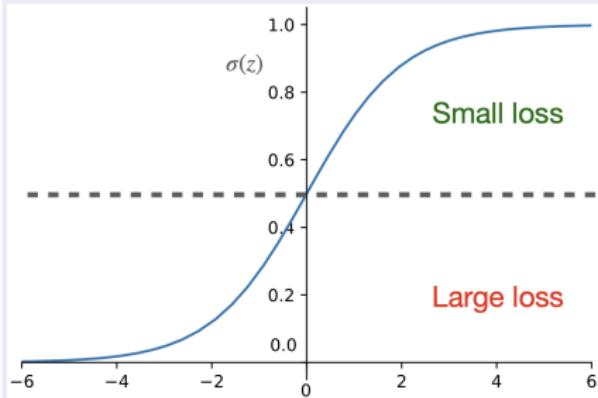
Làm sao để dự đoán một training example thuộc vào lớp 0 hay 1?

Nếu chung ta xem đầu ra của hàm activation trong Logistic regression như một Class membership probability (*xác suất một mẫu dữ liệu thuộc về một lớp nhất định trong một bài toán phân loại*):  $p(y = 0|\mathbf{x})$  và  $p(y = 1|\mathbf{x})$ .

# Cách hoạt động của Logistic regression loss

If the true label is 1

Giả sử rằng chúng ta có true label của một mẫu dữ liệu bằng 1, khi huấn luyện ta mong muốn:

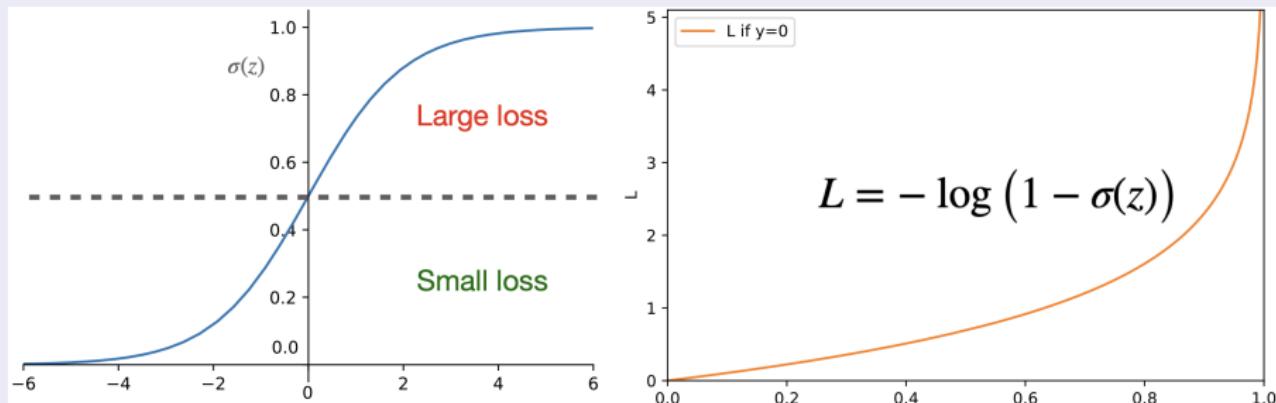


$\sigma(z) > 0.5 \rightarrow$  mô hình sẽ dự đoán đúng thuộc lớp 1. Điều này sẽ làm độ tin cậy  $p(y = 1|x)$  cao  $\rightarrow$  ta thu được một **small loss**.  
 Và ngược lại khi  $\sigma(z) < 0.5 \rightarrow$  dự đoán lớp 0,  $p(y = 1|x)$  thấp  $\rightarrow$  **large loss**

# Cách hoạt động của Logistic regression loss

If the true label is **0**

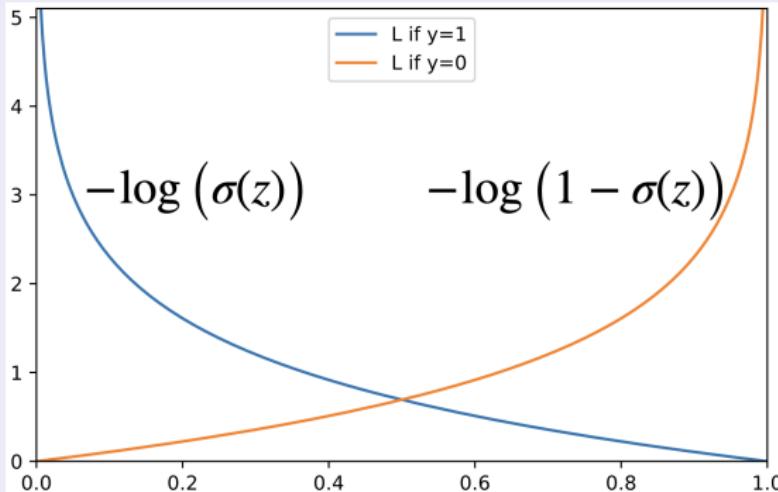
Giả sử rằng chúng ta có true label của một mẫu dữ liệu bằng **0**, khi huấn luyện ta mong muốn:



$\sigma(z) > 0.5 \rightarrow$  mô hình sẽ dự đoán thuộc lớp 1. Điều này sẽ làm độ tin cậy  $p(y = 0|\mathbf{x})$  thấp  $\rightarrow$  ta thu được một **large loss**.

Và ngược lại khi  $\sigma(z) < 0.5 \rightarrow$  dự đoán lớp 0,  $p(y = 0|\mathbf{x})$  cao  $\rightarrow$  **small loss**

Let's now combine these two parts into one single formula



$$L = -y^i \log(\sigma(z)) - (1 - y^i) \log(1 - \sigma(z)) \quad (1)$$

We can then apply 1 to the training examples when we train our logistic regression model.

## One training example

Logistic regression loss function cho 1 mẫu dữ liệu có công thức là:

$$L = -[y^i \log(\sigma(z)) + (1 - y^i) \log(1 - \sigma(z))] \quad (2)$$

## Multiple training examples

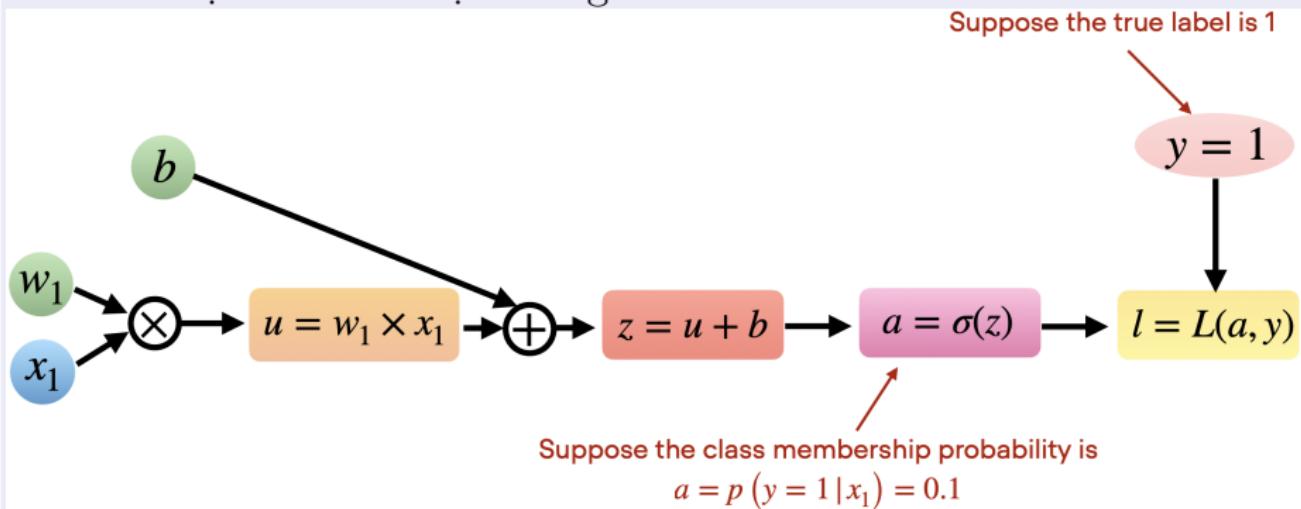
Logistic regression loss function cho tập dữ liệu training có công thức tổng quát là:

$$L = \frac{1}{n} \sum_{i=1}^n -[y^i \log(\sigma(z)) + (1 - y^i) \log(1 - \sigma(z))] \quad (3)$$

Lưu ý:  $\frac{1}{n}$  is a normalization term that helps scale the loss across datasets or batches of different sizes.

# The Logistic Regression Computation Graph

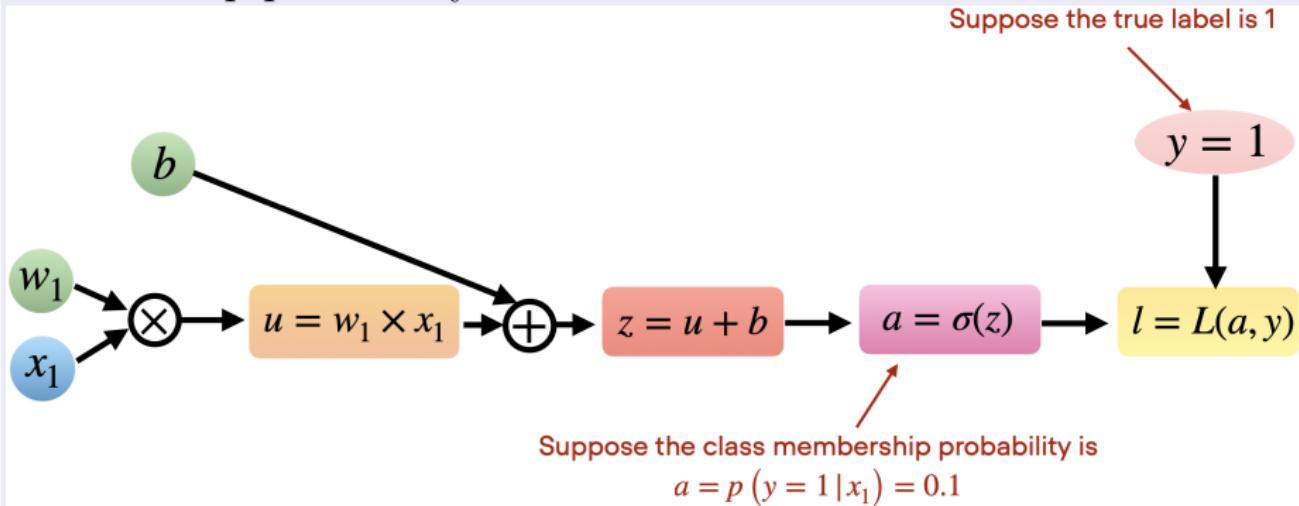
Sơ đồ tính toán cho việc huấn luyện logistic regression trên một mẫu dữ liệu chỉ có 1 đặc trưng:



Trong đó,  $x_1$ : input feature;  $w_1, b$ : weight parameters;  $z$ : weighted sum;  $a$ : logistic regression activation output.

# The Logistic Regression Computation Graph

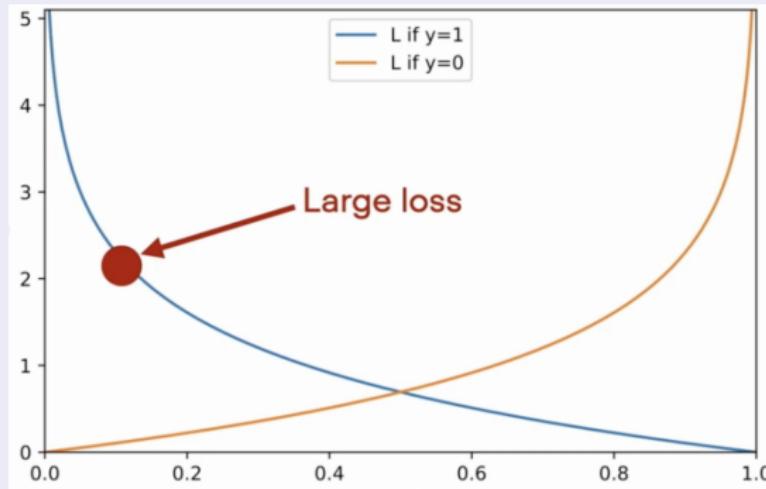
The loss  $l$  is computed between the activation output, the class membership probability and the true class label.



Lưu ý:  $x_1, y$  là các giá trị cho trước.  $w_1, b$  là các tham số mô hình cần phải cập nhật trong quá trình huấn luyện.

## The Logistic Regression Computation Graph

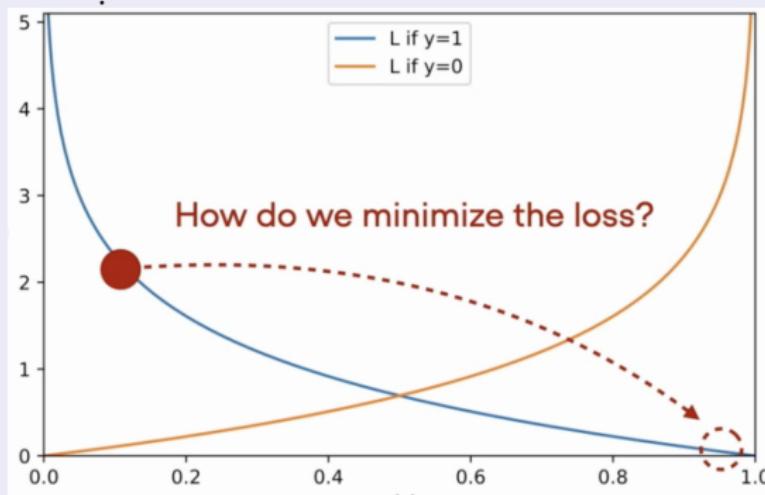
Theo như ví dụ trên, xác suất thành phần của lớp 1:  
 $a = p(y = 1|x_1) = 0.1$ , ta vẽ được đồ thị biểu diễn Large loss như sau



Chúng ta mong muốn giá trị hàm loss càng nhỏ càng tốt để thu được  $\sigma(z)$  tiến về 1.

# The Logistic Regression Computation Graph

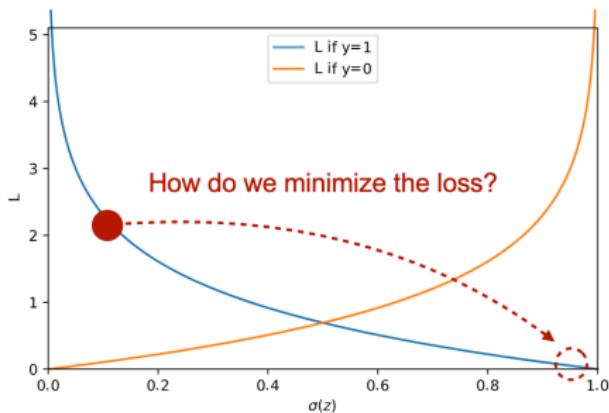
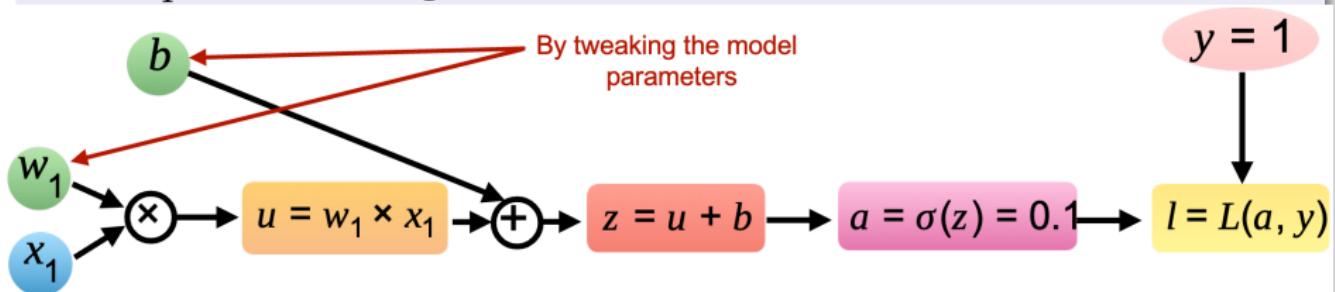
Làm thế nào để cực tiểu hàm Loss?



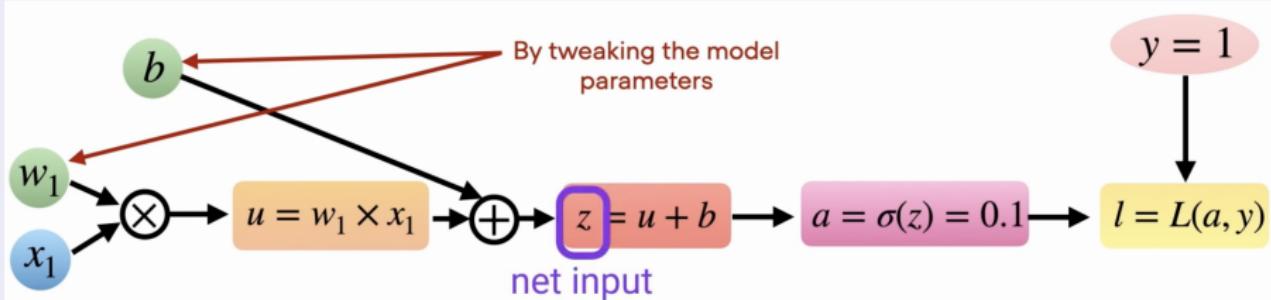
Bắt buộc cập nhật trọng số mô hình  $w_1, b$  sau mỗi lần lặp.

How Do We Change the Model Parameters to Minimize the Loss???

Our goal now is to minimize loss function by tweaking these model parameters  $w_1$  and  $b$ .



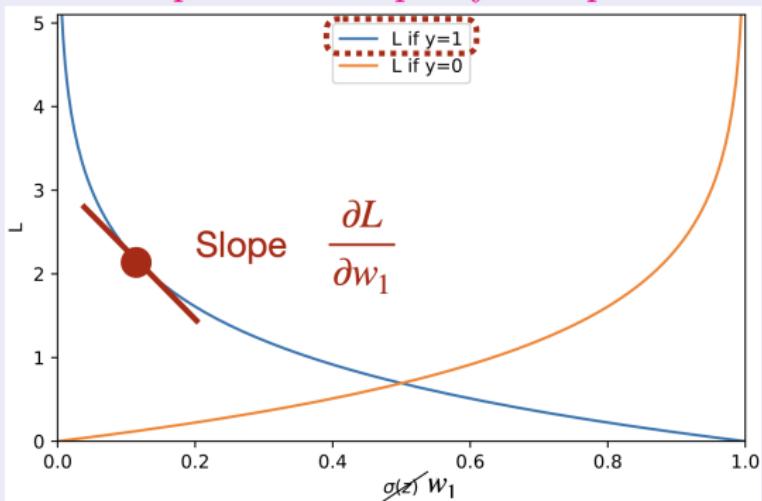
Let's take a look at this loss function as a function of model parameter.



The activation value is computed from the net input  $z$ , and the net input itself consists of  $w_1$ ,  $b$ .

## Changing parameters using the negative slope

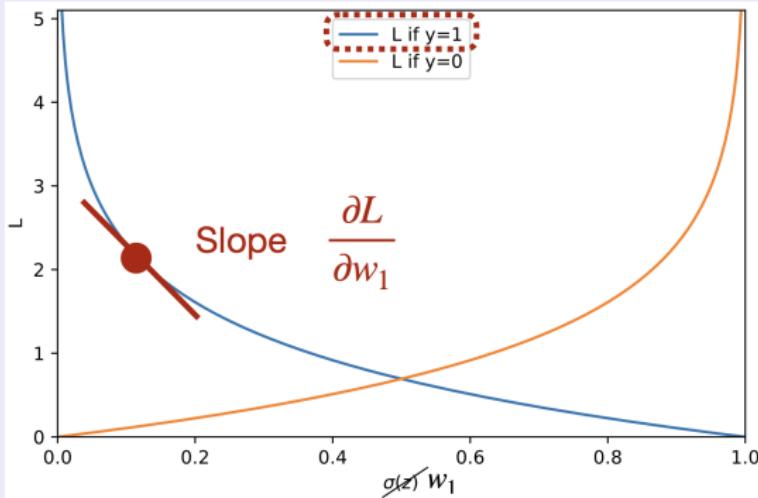
Our goal really is to minimize the loss, *we have to move it to the right hand side here. So for this, we will use a concept from calculus where we compute the slope of this point at this cure.*



Lưu ý: trong trường hợp này chúng ta cố định lại bias unit, chỉ có  $w_1$  thay đổi để cực tiểu hàm Loss, vậy nên ta có thể thay  $\sigma(z)$  như là 1 hàm của  $w_1$ .

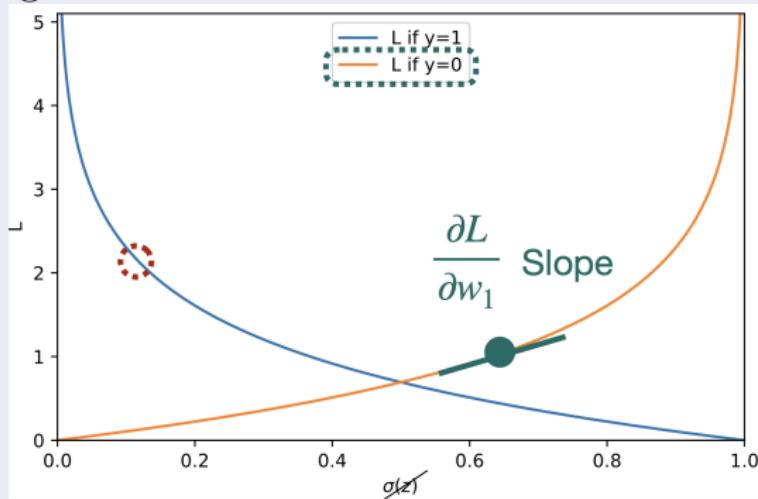
## Changing parameters using the negative slope

Cụ thể, chúng ta sẽ tính đạo hàm riêng theo hướng  $w_1$ .



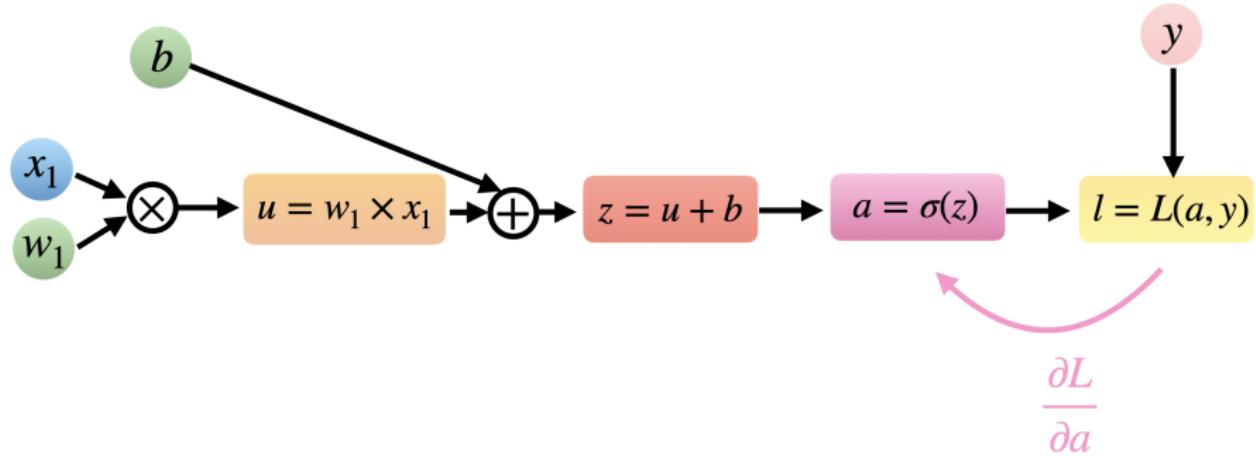
## Changing parameters using the negative slope

Trường hợp ví dụ mẫu có true class label là 0, chúng ta muốn kết quả của hàm activation tiến về càng gần 0 càng tốt. Tính đạo hàm riêng từng tự cho  $w_1$ .



## chain rule

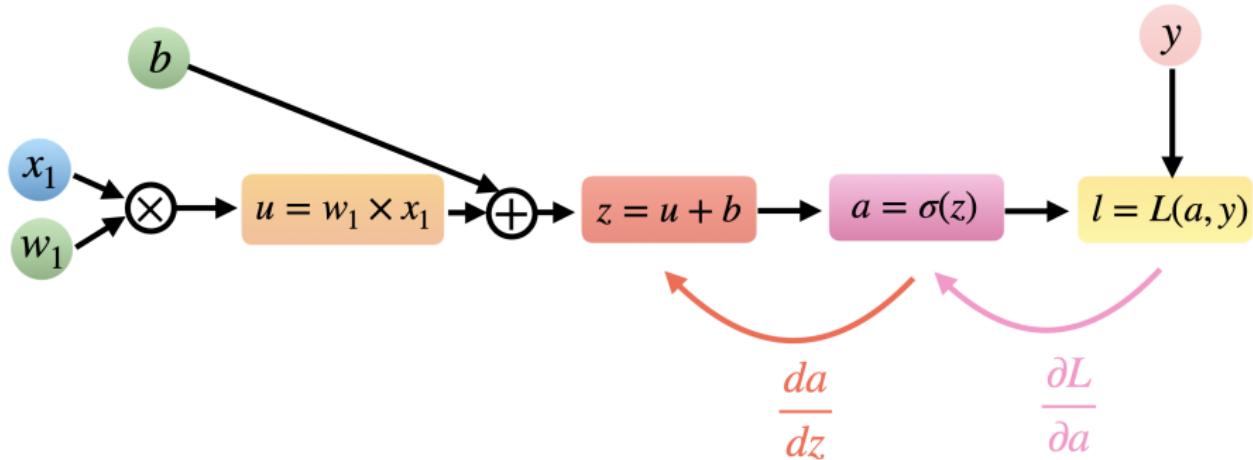
Để tính đạo hàm riêng của hàm Loss theo hướng  $w_1$ , chúng ta sẽ sử dụng quy tắc **chain rule** bởi vì hàm Loss là một hàm hợp.



*we compute the partial derivative of the loss with respect to the activation  $a$*

## chain rule

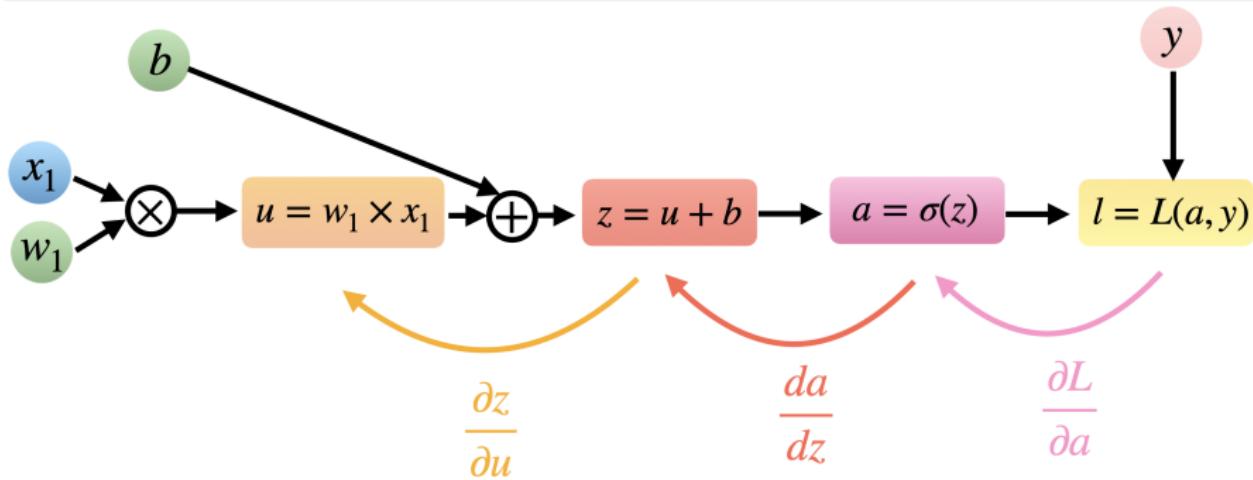
Dể tính đạo hàm riêng của hàm Loss theo hướng  $w_1$ , chúng ta sẽ sử dụng quy tắc **chain rule** bởi vì hàm Loss là một hàm hợp.



*The next we will compute the derivative of  $a$  with respect to the net input  $z$ .*

## chain rule

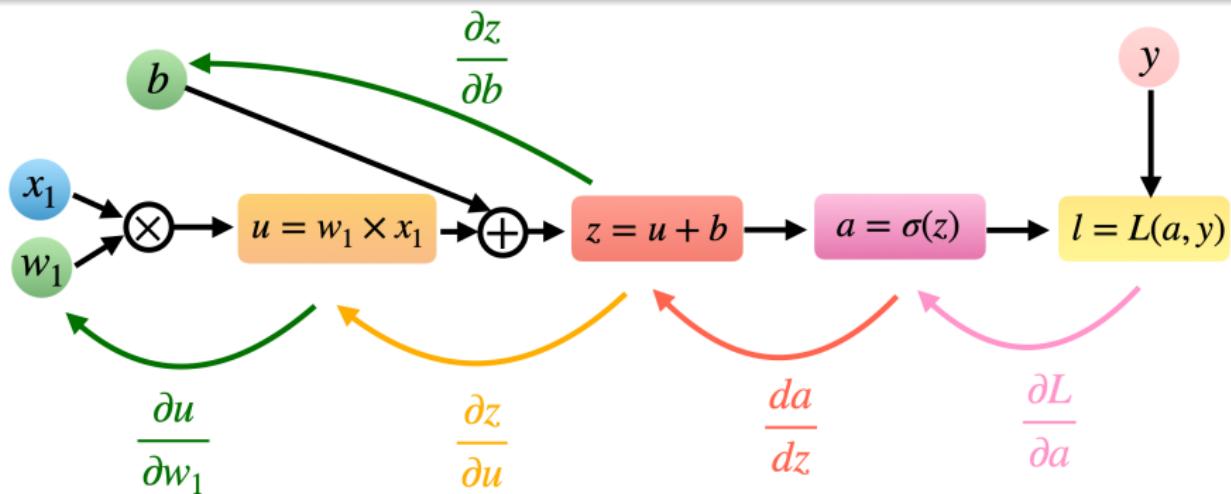
Để tính đạo hàm riêng của hàm Loss theo hướng  $w_1$ , chúng ta sẽ sử dụng quy tắc **chain rule** bởi vì hàm Loss là một hàm hợp.



Next we compute the derivative of the net input  $z$  with respect to  $u$ .

## chain rule

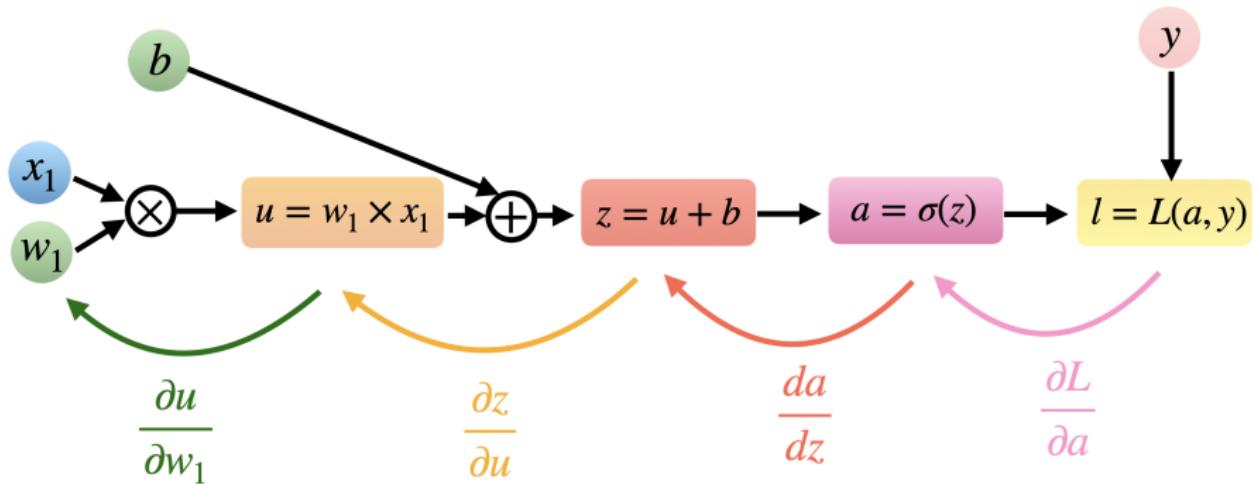
Dể tính đạo hàm riêng của hàm Loss theo hướng  $w_1$ , chúng ta sẽ sử dụng quy tắc **chain rule** bởi vì hàm Loss là một hàm hợp.



Nếu không cố định  $b$ , ở đây chúng ta tính được đạo hàm riêng của  $z$  theo  $b$ .

## chain rule

Để tính đạo hàm riêng của hàm Loss theo hướng  $w_1$ , chúng ta sẽ sử dụng quy tắc **chain rule** bởi vì hàm Loss là một hàm hợp.

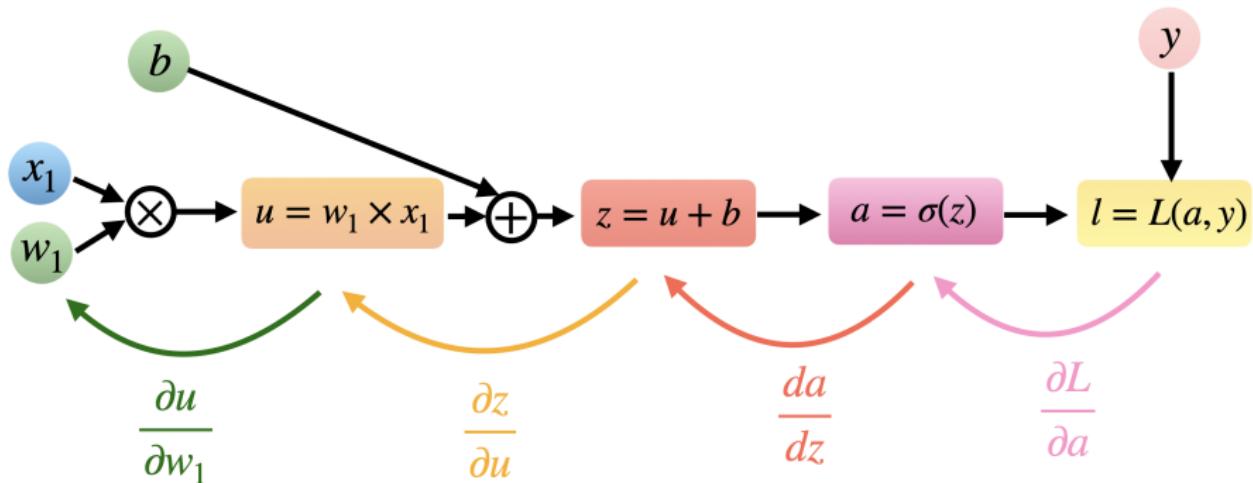


Finally, we compute the partial derivative of  $u$  with respect to  $w_1$ .

## chain rule

Sử dụng **chain rule** ta tính được đạo hàm riêng của hàm Loss theo một trong các tham số của mô hình là  $w_1$  bằng cách kết hợp các đạo hàm riêng:

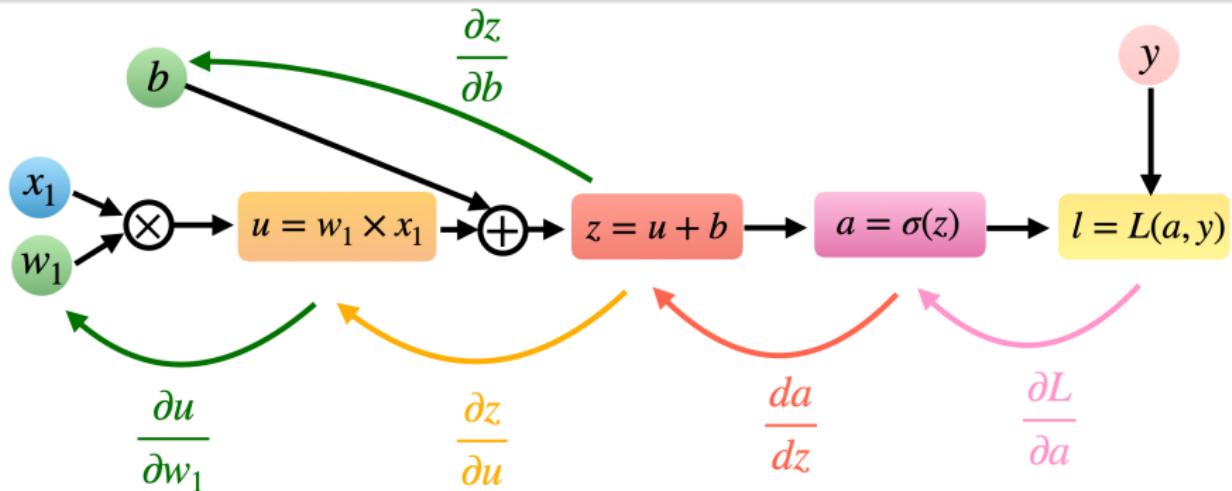
$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial u} \times \frac{\partial u}{\partial w_1}. \quad (4)$$



## chain rule

Tính tương tự đạo hàm riêng của Loss cho  $b$ :

$$\frac{\partial L}{\partial b} = \frac{\partial z}{\partial b} \times \frac{\partial a}{\partial z} \times \frac{\partial L}{\partial a}. \quad (5)$$



PyTorch sẽ hỗ trợ chúng ta tính đạo hàm riêng theo hướng các model parameters.

# Tài liệu tham khảo

-  Sebastian Raschka, Yuxi (Hayden) Liu, Vahid Mirjalili  
Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python (2022). Published by Packt Publishing Ltd, ISBN 978-1-80181-931-2.
-  Sebastian Raschka  
MACHINE LEARNING Q AND AI: 30 Essential Questions and Answers on Machine Learning and AI (2024). ISBN-13: 978-1-7185-0377-9 (ebook).
-  LightningAI  
LightningAI: PyTorch Lightning (2024) .