

# **LOGISTIC REGRESSION**

# Logistic regression

- **Logistic regression** (tạm dịch là Hồi quy logistic) là một thuật toán máy học thường được dùng trong **tác vụ phân lớp** (classification).
- Mục tiêu của logistic regression là ước lượng (estimate) **xác suất** của một điểm dữ liệu **rơi vào một lớp cụ thể**.
- Điểm khác biệt giữa logistic regression và linear regression là **đầu ra của logistic regression được đưa qua một hàm logit** thay vì đưa ra giá trị trực tiếp..

# Mô hình logistic regression

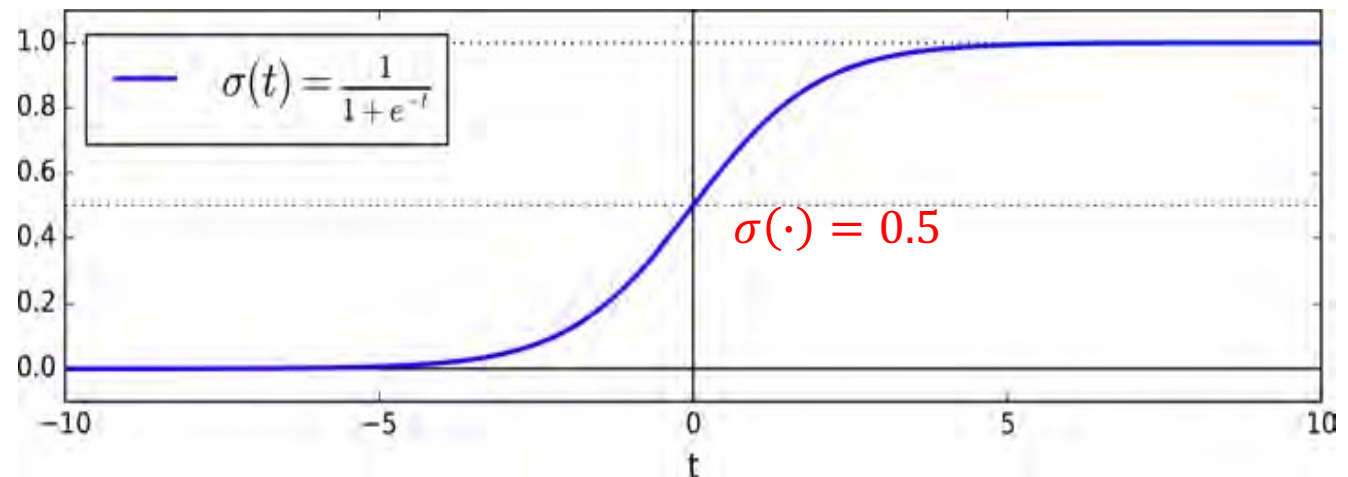
— Mô hình của logistic regression như sau:

$$\hat{p} = \sigma(\theta^T \cdot x)$$

$\sigma = \frac{1}{1+e^{-x}}$  được gọi là hàm số **sigmoid**. Giá trị hàm **sigmoid** trong khoảng  $[0,1]$ .

Giá trị dự đoán  $\hat{p}$  như sau:

$$\hat{p} = \begin{cases} 0 & \text{nếu } \hat{p} < 0.5 \\ 1 & \text{nếu } \hat{p} \geq 0.5 \end{cases}$$



Minh họa hàm sigmoid

# Ý nghĩa của $\hat{p}$

- $\hat{p}$  được gọi là xác suất để 1 điểm dữ liệu  $x$  rơi vào 1 lớp  $c$ .
- Xét bài toán phân lớp nhị phân, sẽ có 2 lớp 0 và 1. Do đó

$$P(y=1 \mid x, \theta) = \hat{p} = \sigma(\theta^T \cdot x)$$

$$P(y=0 \mid x, \theta) = 1 - \hat{p} = 1 - \sigma(\theta^T \cdot x)$$

# Xây dựng hàm mất mát

- Xét bài toán phân lớp nhị phân gồm 2 lớp: 0 và 1.
- Mục tiêu của huấn luyện: tìm ra bộ tham số  $\theta$  để tìm ra xác suất lớn nhất mà một điểm dữ liệu thuộc về một lớp cụ thể.

$$c(\theta) = \begin{cases} -\log(\hat{p}), & \text{if } y = 1 \\ -\log(1 - \hat{p}), & \text{if } y = 0 \end{cases}$$

Ý nghĩa:

Nếu  $y = 1$ , nhưng  $\hat{p} \approx 0 \Rightarrow$  *loss sẽ rất lớn*.

Nếu  $y = 1$  và  $\hat{p} \approx 1 \Rightarrow$  *loss sẽ rất nhỏ*.

(tương tự đối với  $y = 0$ )

# Xây dựng hàm mất mát

– Hàm mất mát tổng cộng cho **cả 2 lớp** được mô tả như sau:

$$\mathcal{L}(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})]$$

**Mục tiêu:**  $\theta^* = \text{argmin}_{\theta}(\mathcal{L}(\theta))$

Tính đạo hàm của  $\mathcal{L}$  theo  $\theta$

$$\frac{\partial \mathcal{L}}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^m (\sigma(\theta^T x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Để tìm cực trị của  $\mathcal{L} \rightarrow$  sử dụng **Gradient descent** (sẽ học sau).

Ký hiệu (notation):

- $x^{(i)}$ : vector đặc trưng điểm dữ liệu thứ  $i$  trong bộ dữ liệu.
- $m$ : số lượng điểm dữ liệu.
- $y^{(i)}$ : giá trị của điểm dữ liệu thứ  $i$  trong bộ dữ liệu.
- $j$ : đặc trưng thứ  $j$  trong bộ dữ liệu.

# Tìm cực trị của hàm mất mát

— Sử dụng Gradient Descent:

$W := \boldsymbol{\theta}_0$       // Khởi tạo trọng số

Repeat {

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha^* \frac{d\mathcal{L}(\boldsymbol{\theta}, b)}{d\boldsymbol{\theta}}$$

}

— Vector gradient của  $\boldsymbol{\theta}$  được tính như sau:

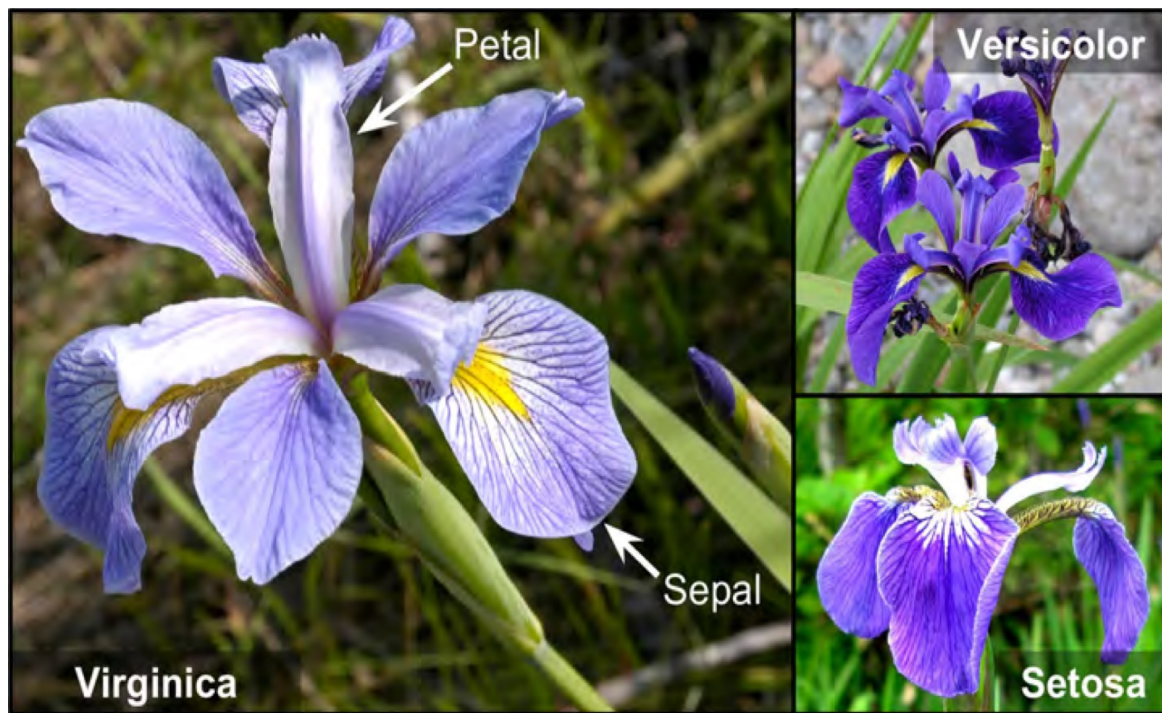
$$\frac{d\mathcal{L}(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \frac{1}{m} X^T (\hat{y} - y)$$

# Minh họa

- Bộ dataset **Iris** (<https://archive.ics.uci.edu/ml/datasets/iris>) chứa thông tin về độ dài (length) và độ rộng (width) của **cánh hoa** (petal) và **đài hoa** (sepal) của **hơn 150 loài hoa iris**.
- Bài toán đặt ra: dựa vào thuộc tính độ rộng của cánh hoa (sepal width), **dự đoán xem hoa đó có thuộc loài Iris-Virginica hay không?**
  - + Đầu vào: *petal width features*.
  - + Đầu ra:
    - **1** - thuộc loài Iris-Virginica
    - **0** – không thuộc loài Iris-Virginica



# Iris dataset



Iris

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5.0	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3.0	1.4	0.1	Iris-setosa
14	4.3	3.0	1.1	0.1	Iris-setosa
15	5.8	4.0	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa

# Chuẩn bị dữ liệu

- Bộ dữ liệu Iris được hỗ trợ sẵn bởi sklearn.

```
from sklearn import datasets  
iris = datasets.load_iris()
```

- Lấy thuộc tính petal width:

```
X = iris["data"][:, 3:]
```

- Thuộc tính đích: giá trị 0 nếu như không phải là Iris-Virginica và giá trị 1 nếu như là Iris-Virginica.

```
y = (iris["target"] == 2).astype(np.int)
```

# Xây dựng mô hình

- Xây dựng mô hình Logistic Regression từ dữ liệu huấn luyện  $(X, y)$ .

```
from sklearn.linear_model import LogisticRegression
```

```
log_reg = LogisticRegression()
```

```
log_reg.fit(X, y)
```

# Dự đoán dữ liệu mới $X_{\text{new}}$

— Tạo dữ liệu mới  $X_{\text{new}}$ :

```
X_new = np.linspace(0, 3, 1000).reshape(-1, 1)
```

— Dự đoán giá trị cho dữ liệu mới:

```
y_proba = log_reg.predict_proba(X_new)
```

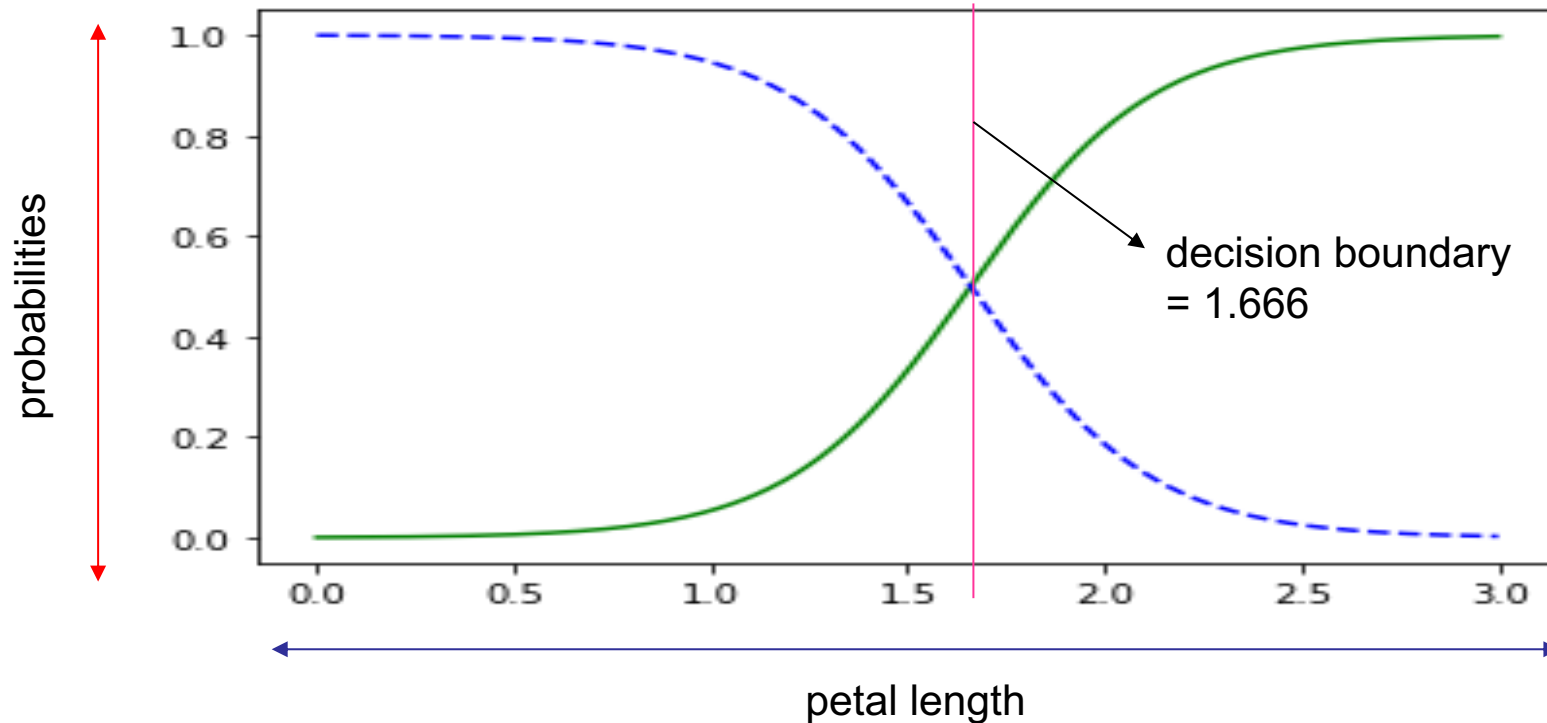
— Trực quan hoá dữ liệu:

```
plt.plot(X_new, y_proba[:, 1], "g-", label="Iris-Virginica")  
plt.plot(X_new, y_proba[:, 0], "b--", label="Not Iris-Virginica")
```

# Decision boundary

— Lấy giá trị decision\_boundary:

```
decision_boundary = X_new[y_proba[:, 1] >= 0.5][0]
```



- Nếu *petal\_length* trên 1.6, mô hình sẽ dự đoán là **Iris-Virginica**.
- Nếu *petal\_length* dưới 1.6, mô hình sẽ dự đoán không phải là **Iris-Virginica**

# Dự đoán

— Dự đoán nhãn cho 2 giá trị petal\_length: 1.7 và 1.5

```
log_reg.predict([[1.7], [1.5]])
```

— Kết quả:

```
array([1, 0])
```

➤ Giá trị 1.7 được dự đoán là Iris-Virginica.

➤ Giá trị 1.5 được dự đoán không phải là Iris-Virginica.

# **SOFTMAX REGRESSION**

# Multi-class classification

- Logistic regression thường được dùng cho bài toán phân lớp nhị phân (binary classification).
- Nếu trên 2 lớp, LR vẫn có thể áp dụng bằng các chiến lược như **Một với tất cả nhãn còn lại (OvR)**.
  - + Nhược điểm: n class cần n bộ phân lớp → khá tốn chi phí !!
- Để khắc phục nhược điểm trên, **Softmax regression** được sử dụng.
  - + **Ý tưởng**: ước lượng xác suất một điểm dữ liệu thuộc về một lớp trên k lớp ( $k > 2$ )



# Softmax regression

— Công thức softmax regression:

$$\hat{p}_k = \frac{e^{\sigma(\theta_k^T \cdot \mathbf{x})}}{\sum_{j=1}^K e^{\sigma(\theta_j^T \cdot \mathbf{x})}}$$

- K: số lượng lớp (nhãn)
- k: lớp cụ thể trong tập các lớp (nhãn)
- $\sigma(\theta_k^T \cdot \mathbf{x})$ : xác suất ước lượng của 1 điểm dữ liệu thuộc về

— Giá trị dự đoán: nhãn (label) có **xác suất ước lượng cao nhất**.

$$\hat{y} = \text{argmax}_k (\theta_k^T \cdot \mathbf{x})$$

# Xây dựng hàm mất mát

— Hàm mất mát cho softmax regression:

$$\mathcal{L}(\theta) = \frac{-1}{m} \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(\hat{p}_k^{(i)})$$

$y_k^{(i)}$ : xác suất **điểm dữ liệu thứ (i)** thuộc về **lớp k**.

— Tính đạo hàm của  $\mathcal{L}$  theo  $\theta$ :

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{1}{m} \sum_{i=1}^m \left( \hat{p}_k^{(i)} - y_k^{(i)} \right) \mathbf{x}^{(i)}$$

Tối ưu  $\mathcal{L} \rightarrow$  sử dụng Gradient descent.

# Minh hoạ với dữ liệu Iris

- Lấy thuộc tính *petal\_length* và *petal width*:

```
X = iris["data"][:, (2, 3)]
```

- Thuộc tính đích: loài hoa iris tương ứng, thuộc 1 trong 3 giá trị như sau:

- + 0: Iris-setosa

- + 1: Iris-versicolor

- + 2: Iris-Virginica

```
y = iris["target"]
```

# Xây dựng mô hình

— Sử dụng Logistic regression với các tham số sau:

+ `multi_class="multinomial"`

+ `solver="lbfgs"`

+ `C=10`

```
softmax_reg = LogisticRegression(  
    multi_class="multinomial",  
    solver="lbfgs",  
    C=10)
```

```
softmax_reg.fit(X, y)
```

# Dự đoán

- Dự đoán nhãn cho dữ liệu có giá trị `petal_length = 5` và `petal_width = 2`.
- Giá trị xác suất dự đoán ứng với từng nhãn:

```
softmax_reg.predict_proba([[5, 2]])
```

➤ Kết quả: `array([[6.38014896e-07, 5.74929995e-02, 9.42506362e-01]])`

*lớp 0*

*lớp 1*

*lớp 2*

Lớp dự đoán: **Iris-Virginica (lớp 2)**

# TÀI LIỆU THAM KHẢO

Chương 4 của sách: *Hands-on Machine Learning with ScikitLearn, Keras & TensorFlow*.