BÀI GIẢNG HỌC MÁY

Bài 5-Tiết 1: Mô hình hồi quy Logistic

Trình bày: Ths Phạm Việt Anh

Viện Công nghệ HaUI
Trường Đại học Công nghiệp Hà Nội

Nội dung của bài học

1. Giới thiệu về mô hình hồi quy Logistic
2. Hàm mất mát trong mô hình Logistic
3. Một số nhận xét về mô hình Logistic
4. Thuật toán Gradient descent

Giới thiệu về mô hình Logistic

- Hồi quy Logistic là một giải thuật thuộc lĩnh vực thống kê.
- Là giải thuật phân loại được sử dụng để dự đoán xác suất.

	LotArea	TotalBsmtSF	Bedroom	SalePrice
0	8450	856	3	208500
1	9600	1262	3	181500
2	11250	920	3	223500
3	9550	756	3	140000
4	14260	1145	4	250000

Biến liên tục → Mô hình hồi quy tuyến tính

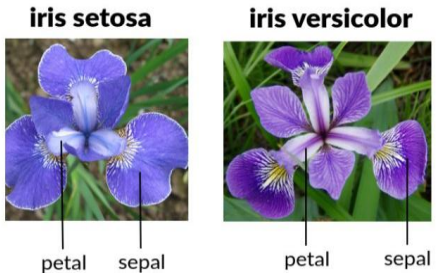
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica

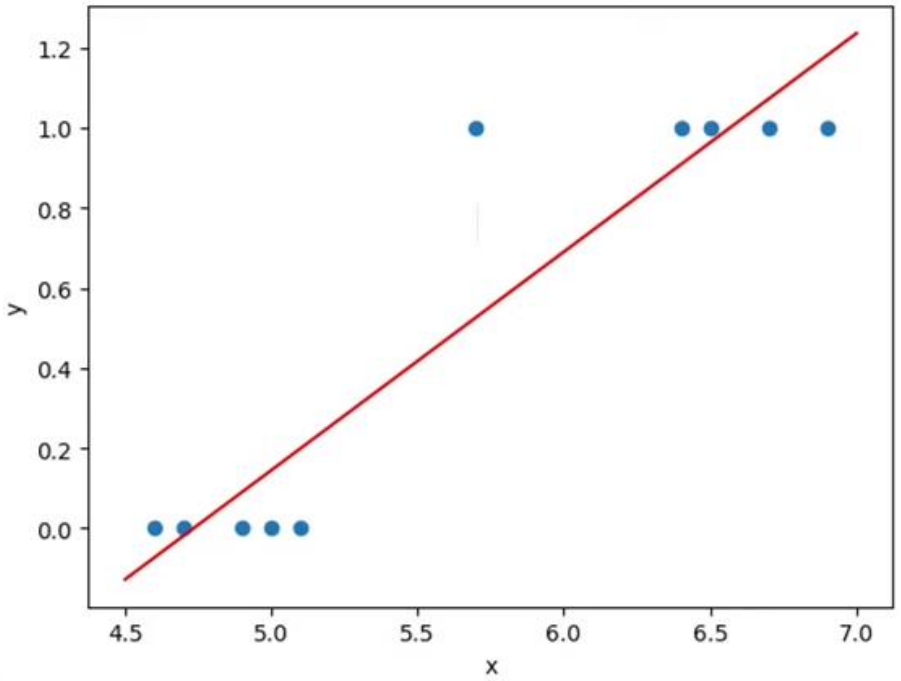
Biến rời rạc → Mô hình hồi quy Logistic

Giới thiệu về mô hình Logistic

❑ **Xét bộ dữ liệu Iris với một biến số:**

Objects	Sepal length	Class
x_1	5.1	0
x_2	4.9	0
x_3	4.7	0
x_4	4.6	0
x_5	5.0	0
x_6	6.4	1
x_7	6.9	1
x_8	6.5	1
x_9	6.7	1
x_{10}	5.7	1

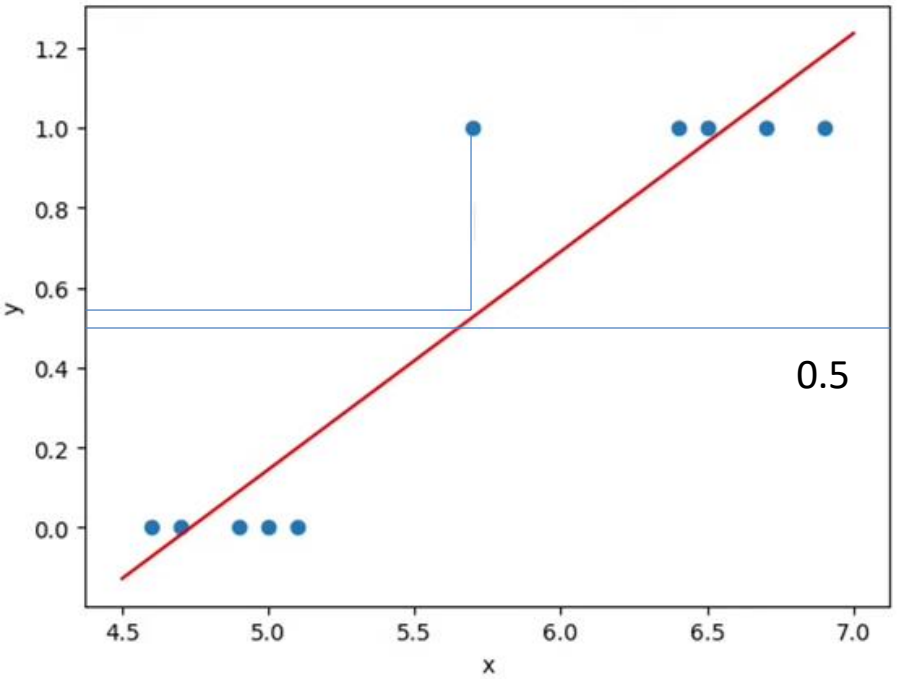


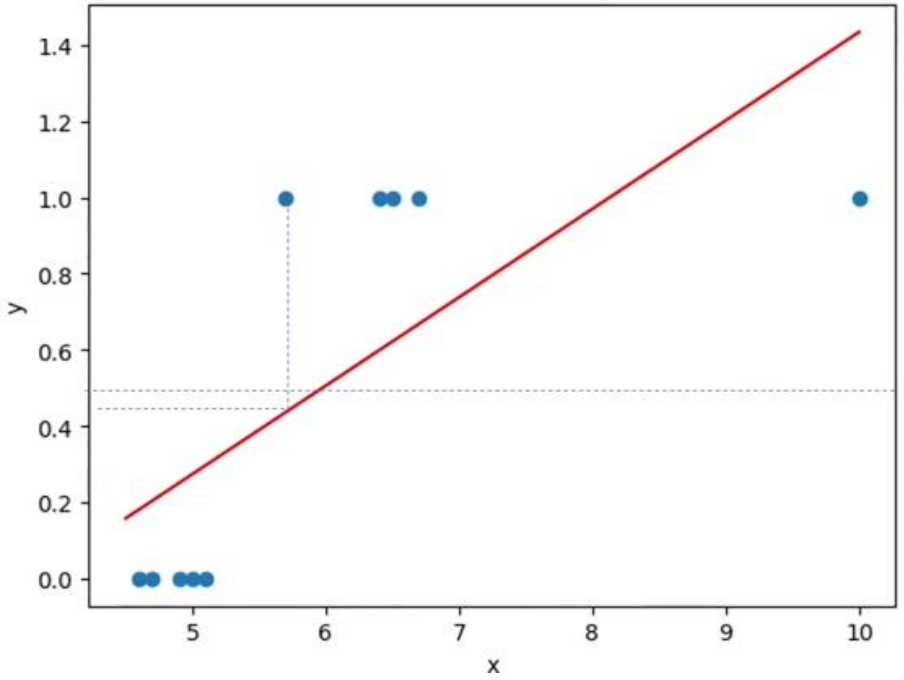


Giới thiệu về mô hình Logistic

❑ **Nếu sử dụng HQTT**:

✓ **Không có một ngưỡng cụ thể mà phải thay đổi ngưỡng liên tục.**





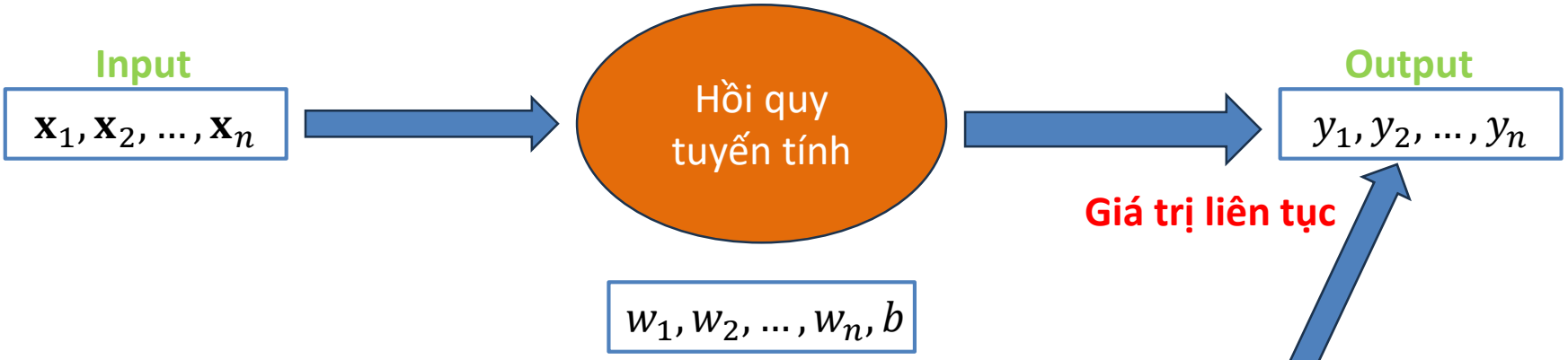
✓ **Dữ liệu có thể vượt ngoài ranh giới.**

✓ **Nhậy cảm với ngoại lai và không phù hợp khi giá trị input không có quan hệ tuyến tính với giá trị output**

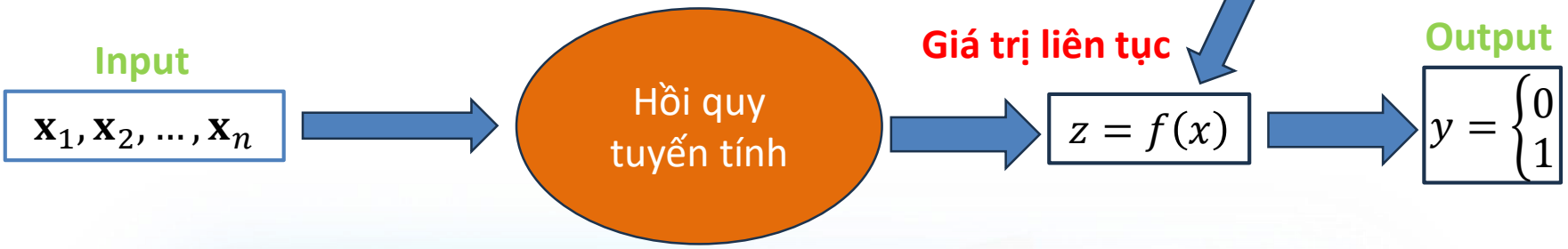
Giới thiệu về mô hình Logistic

❑ So sánh giữa hồi quy tuyến tính và hồi quy Logistic

Hồi quy tuyến tính: $y = f(x) = w_1x_1 + w_1x_1 + ... + w_mx_m + b$



Hồi quy Logistic:

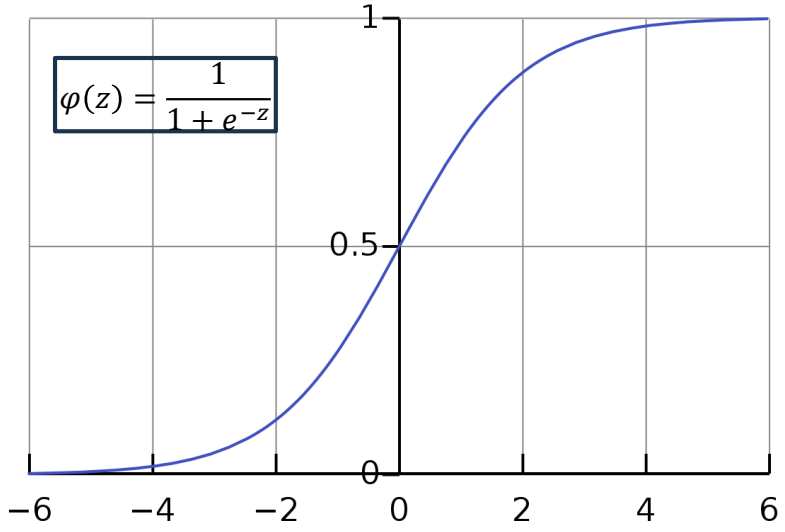


❑ Như vậy, hàm *z* chỉ như một bước trung gian để tìm đầu ra

Giới thiệu về mô hình Logistic

❑ Hàm Sigmoid:

$$\varphi(z) = \frac{1}{1+e^{-z}}$$

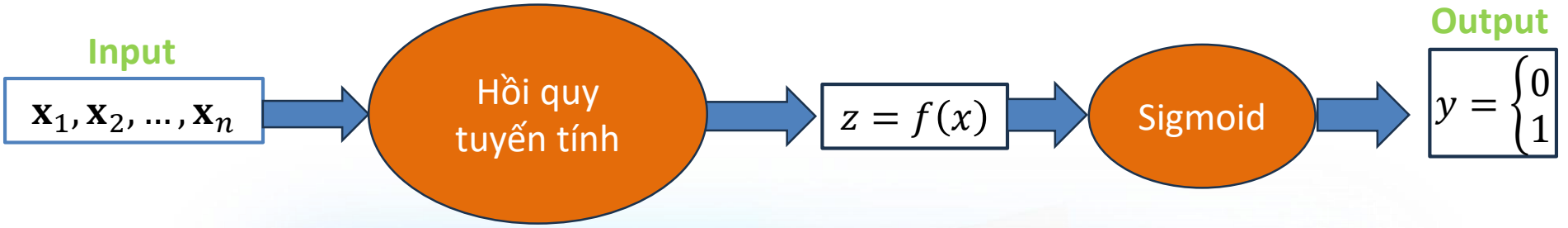


Trong đó:

- $\varphi(z)$ biểu diễn hàm sigmoid với đầu ra nằm trong khoảng từ 0 tới 1.
- Giá trị e là logarit tự nhiên (~2.717).
- z là hàm đầu vào $(-\infty, +\infty)$.

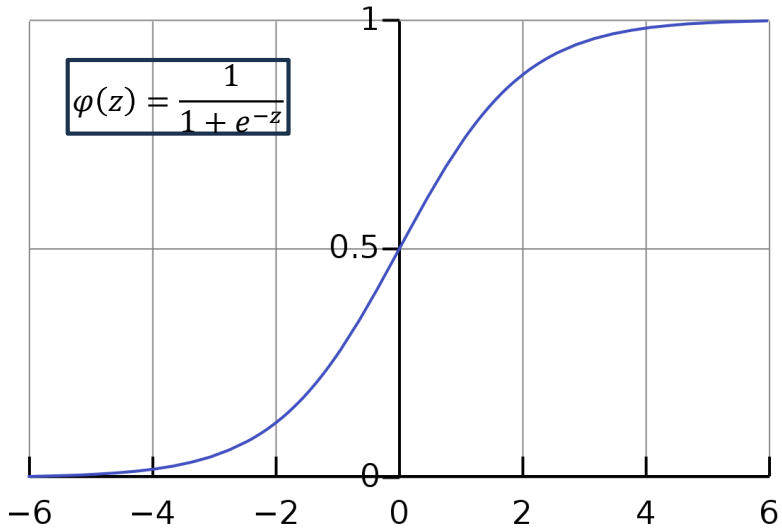
Ý nghĩa:

- Chuyển một giá trị với biến thực
- Giá trị e là logarit tự nhiên (~2.717).
- z là hàm đầu vào $(-\infty, +\infty)$.
- Có đạo hàm đơn giản cho phép tối ưu mô hình dễ dàng.



Giới thiệu về mô hình Logistic

- **Hàm Sigmoid:** $\varphi(z) = \dfrac{1}{1+e^{-z}}$



Chứng minh: $\varphi(x) = \varphi(x)\big(1-\varphi(x)\big)$

Giới thiệu về mô hình Logistic

Hồi quy tuyến tính: $y = f(x) = w_1x_1 + w_1x_1 + ... + w_mx_m + b$

Hồi quy Logistic: $P(y = 1|x) = \varphi(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+exp^{-(w_1x_1+w_1x_1+...+w_mx_m+b)}}$

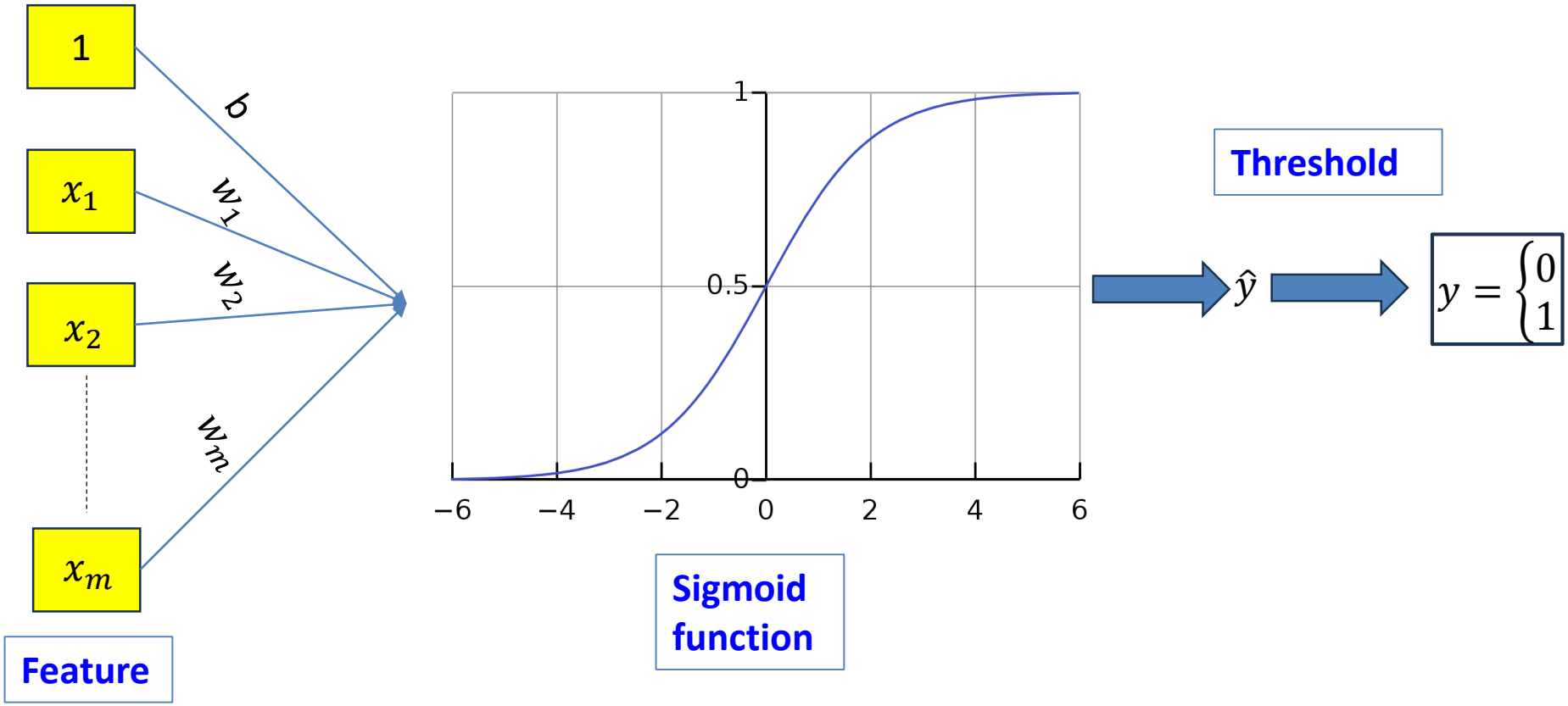
Xác suất để output y có giá trị bằng 1 được đưa ra bởi đặc trưng x

Áp dụng với ngưỡng (threshold = 0.5)

- $\hat{y} = P \geq 0.5$, lớp dương (positive class)
- $\hat{y} = P < 0.5$, lớp âm (negative class)

❑ **So với HQTT, hồi quy Logistic cũng tìm kiếm các trọng số w và b, nhưng chỉ khác và hàm mất mát (loss function). Hàm tối ưu khác nhau.**

Giới thiệu về mô hình Logistic



Hàm mất mát mô hình Logistic

❑ Hàm mất mát:

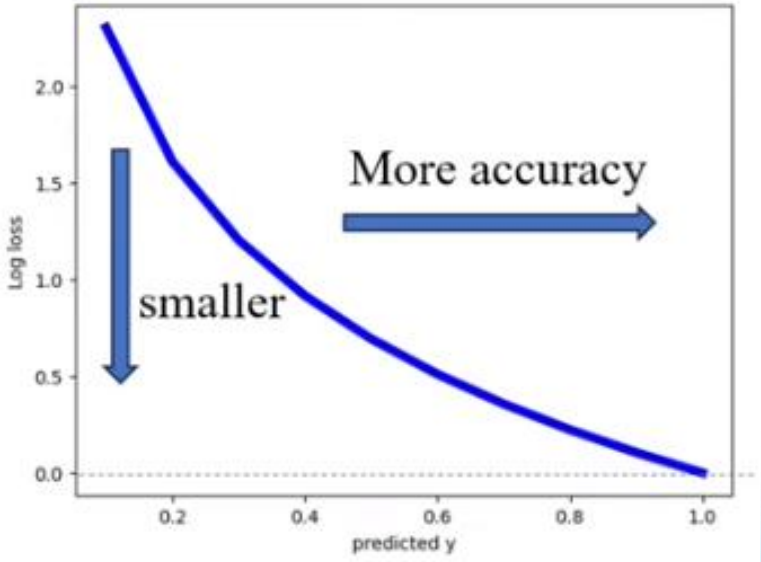
$$\mathcal{L}(\mathbf{w}, \boldsymbol{x_i}, y_i) = -[y_i * log\hat{y}_i + (1 - y_i) * log(1 - \hat{y}_i)]$$

Trong đó:

- $\hat{y}_i$ là giá trị dự đoán (xác suất)
- y_i là giá trị thực (nhãn)
- n là số lượng đối tượng

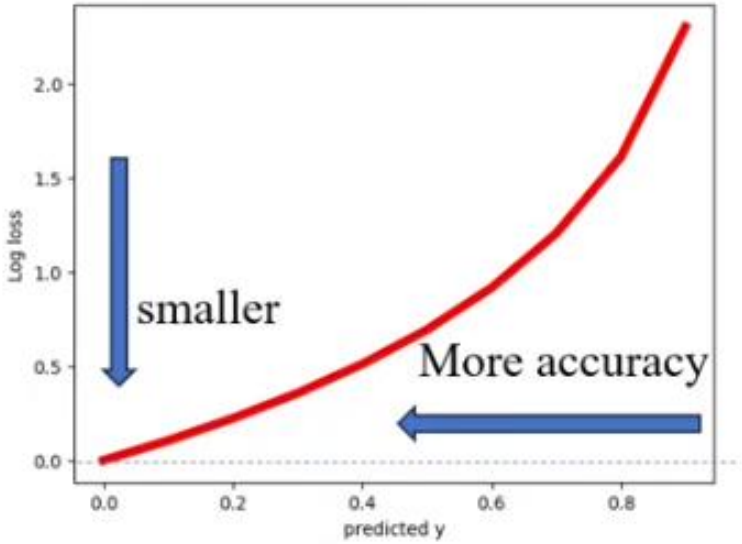
$\boldsymbol{y_i = 1}$

$\Rightarrow \mathcal{L}(\mathbf{w}, \boldsymbol{x_i}, y_i) = -log\hat{y}_i$



$\boldsymbol{y_i = 0}$

$\Rightarrow \mathcal{L}(\mathbf{w}, \boldsymbol{x_i}, y_i) = log(1 - \hat{y}_i)$



Nhận xét về mô hình

Ưu điểm

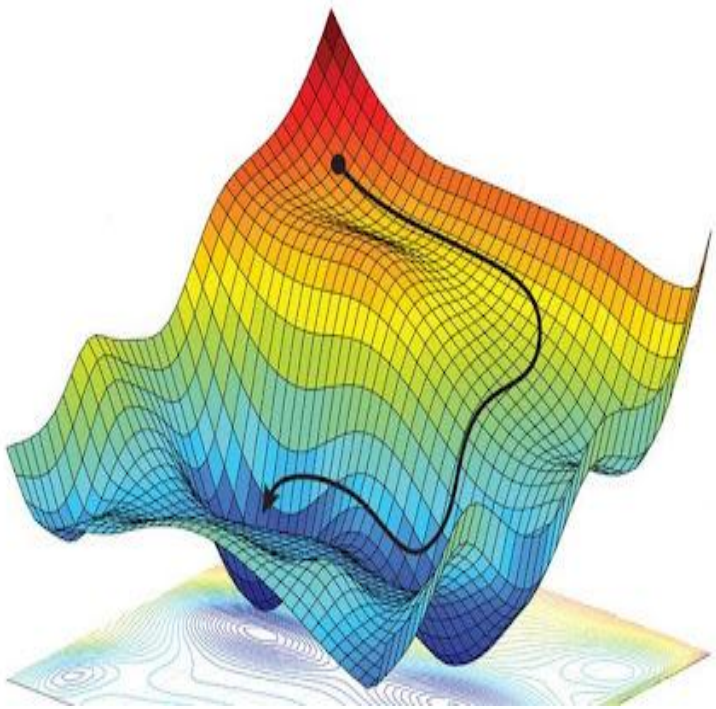
- Đơn giản và dễ dàng tiếp cận
- Hiệu quả với các bài toán phân loại, đặc biệt phân loại nhị phân.
- Hạn chế được overfiting khi sử dụng các hàm phạt.
- Sử dụng trong cả bài toán nhiều lớp.

Nhược điểm

- Không tốt với dữ liệu không cân bằng.
- Kém hiệu quả hơn với các thuật toán khác như SVM, Random forest
- Phải sử dụng qua mô hình hồi quy tuyến tính.

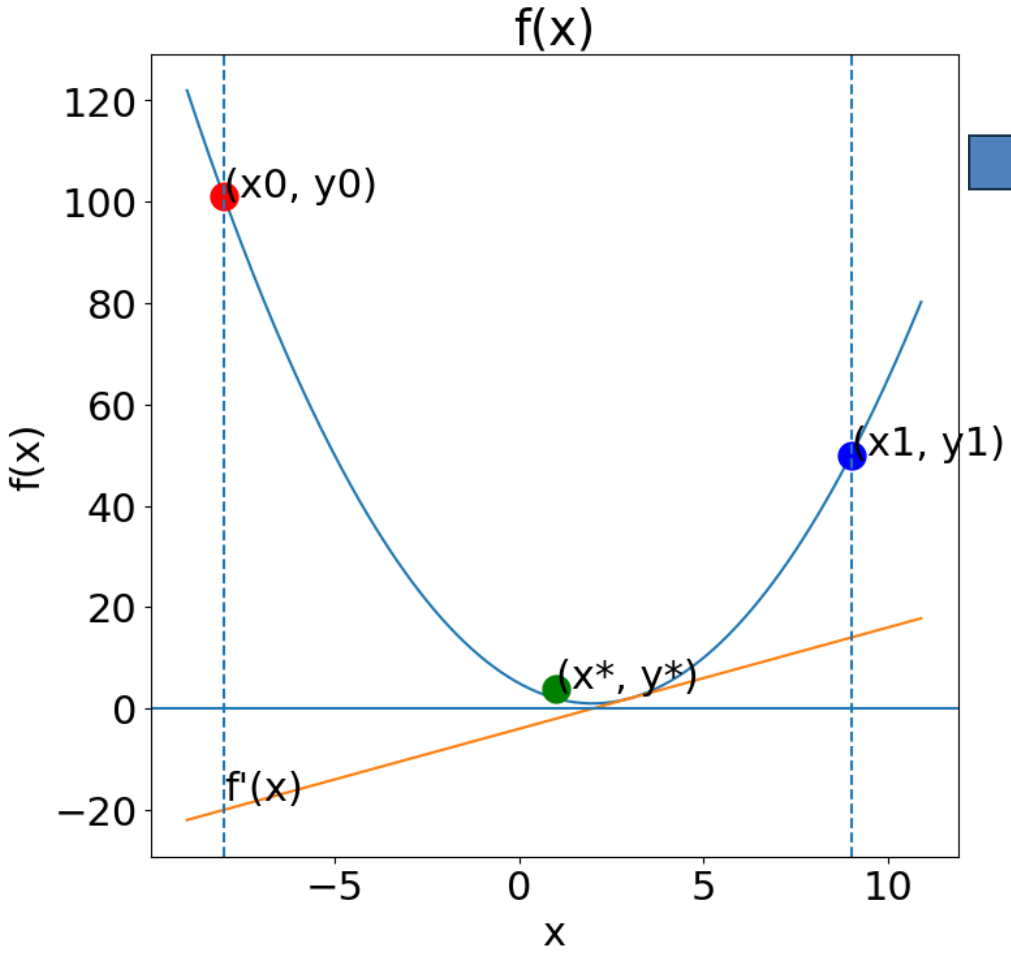
Thuật toán Gradient descent

- Trong thực tế, việc tìm kiếm nghiệm toàn cục của một số mô hình học máy là rất khó.
- Gradient descent: Là một kỹ thuật quan trọng trong học máy và học sâu nhằm tìm cực trị địa phương của mọi hàm số.
- Hạn chế của phương pháp là chỉ tìm được nghiệm gần đúng và không đảm bảo sự chắc chắn so với nghiệm toàn cục.



Thuật toán Gradient descent

Ví dụ: Tìm cực trị của hàm số $f(x) = x^2 - 4x + 5$



Cần di chuyển ngược chiều đạo hàm để tiến gần hơn tới cực trị.

Dùng một phép toán lặp để *tiến dần* đến điểm cần tìm, tức đến khi đạo hàm gần với 0.

$$x_{new} := x_0 - \alpha \nabla_x f(x)$$

Trong đó:

- α được gọi là tốc độ học
- x_0 là điểm bắt đầu

Thuật toán Gradient descent

- Step 1: Khởi tạo giá trị $x = x_0$ tùy ý
- Step 2: Gán lại $x = x - \alpha \nabla_x f(x)$
- Step 3: Tính lại $f(x)$, nếu $f(x)$ đủ nhỏ theo một ngưỡng cho trước thì thuật toán dừng lại

Ví dụ: Tìm cực trị của hàm số $f(x) = x^3 - 5x^2 + 5$

Giải:

- Khởi tạo $x = 1$, chọn $\alpha = 0.01$, $\nabla_x f(x) = 3x^2 - 10x$
- Lần 1: Gán lại $x = 1 - 0.01(3 * 1^2 - 10 * 1) = 1.07 \Longrightarrow f(x) = 0.5$
- Lần 2: Gán lại $x = 1.07 - 0.01(3 * 1.07^2 - 10 * 1.07) = 1.073 \Longrightarrow f(x) = 0.482$
- Lần n...

Nghiệm của hồi quy Logistic

- Cập nhật nghiệm trên từng điểm dữ liệu $(\mathbf{x_i}, y_i)$ để tìm ra nghiệm của hồi quy Logistic.

$$\boldsymbol{w} := \mathbf{w} - \alpha \frac{\delta \mathcal{L}(\mathbf{w}, \boldsymbol{x_i}, y_i)}{\delta \mathbf{w}}$$

- Các w_i được cập nhật đồng thời.

$$\frac{\delta \mathcal{L}(\mathbf{w}, \boldsymbol{x_i}, y_i)}{\delta \mathbf{w}} = -\left[y_i \frac{\delta log \hat{y}_i}{\delta \mathbf{w}} + (1 - \hat{y}_i) \frac{\delta log(1 - \hat{y}_i)}{\delta \mathbf{w}} \right]$$

$$= -\left[y_i \frac{\delta log \hat{y}_i}{\delta \hat{y}_i} \frac{\delta \hat{y}_i}{\delta \mathbf{w}} + (1 - y_i) \frac{\delta log(1 - \hat{y}_i)}{\delta \hat{y}_i} \frac{\delta \hat{y}_i}{\delta \mathbf{w}} \right]$$

$$= -\left[y_i \frac{1}{\hat{y}_i} - (1 - y_i) \frac{1}{(1 - \hat{y}_i)} \right] \frac{\delta \hat{y}_i}{\delta \mathbf{w}}$$

$$= -\left[\frac{(y_i - \hat{y}_i)}{\hat{y}_i (1 - \hat{y}_i)} \right] \frac{\delta \hat{y}_i}{\delta \mathbf{w}} = \mathbf{x_i} (\hat{y}_i - y_i)$$

$$\boxed{\mathbf{w} := \mathbf{w} - \alpha \mathbf{x_i} (\hat{y}_i - y_i)}$$

Tốc độ học (learning rate)

- Nếu learning rate quá nhỏ, thuật toán sẽ phải thực hiện nhiều bước để hội tụ và sẽ mất nhiều thời gian.
- Tuy nhiên nếu learning rate quá lớn sẽ khiến thuật toán đi qua cực tiểu, và vượt hẳn ra ngoài khiến thuật toán không thể hội tụ được.

Cài đặt thuật toán

- **Cho bộ dữ liệu sau:**

Số buổi đi học	Thời gian làm bài (h)	Kết quả
6	2	0
8	1	0
9	3	0
10	7	1
9	9	1
5	3	0
8	12	1

- **Xây dựng chương trình tìm các trọng số theo mô hình Hồi quy Logistic dựa trên kỹ thuật Gradient descent.**

Cài đặt thuật toán

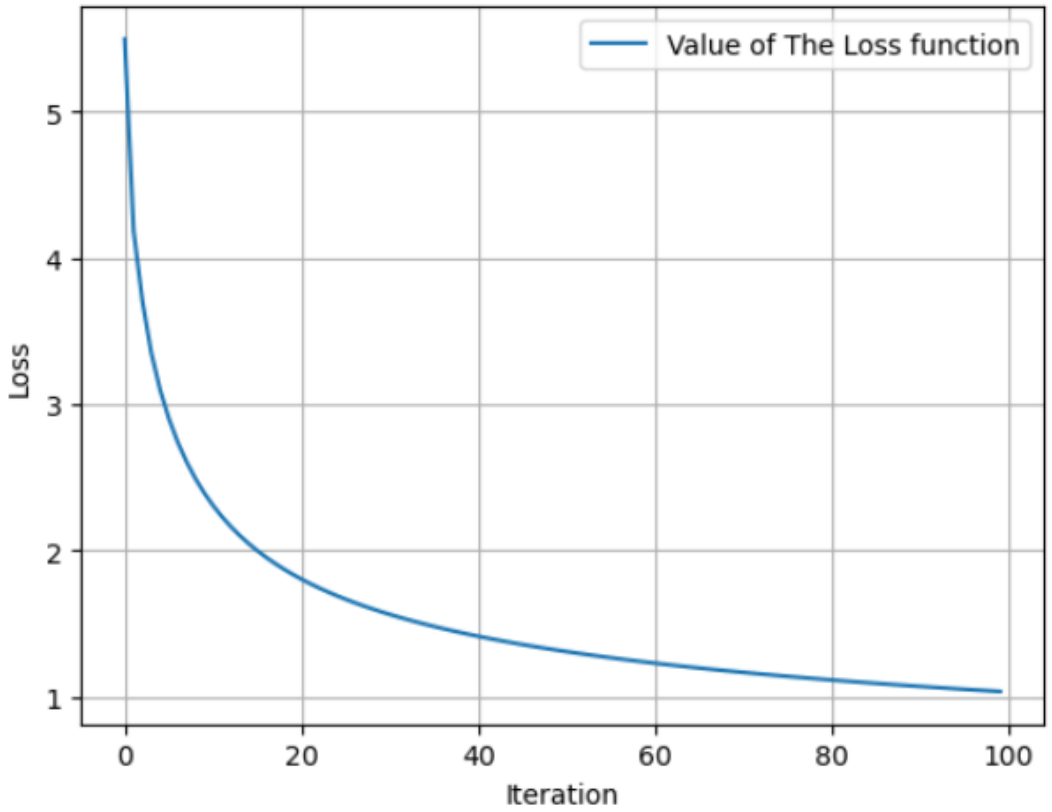
In [1]:
```
# Chuẩn bị một số thư viện cần thiết
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
# Chuẩn bị dữ liệu huấn luyện
x1 = np.array([[6, 8, 9, 10, 9, 5, 8]]).T #n x 1
x2 = np.array([[2, 1, 3, 7, 9, 3, 12]]).T #n x 1

X = np.concatenate((x1,x2), axis = 1)
one = np.ones((X.shape[0],1))
X = np.concatenate((one, X), axis = 1)

y = np.array([[0, 0, 0, 1, 1, 0, 1]]).T #n x 1
```

In [3]:
```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# Thiết lập tham số và mảng chứa giá trị hàm loss
NUMITERATION = 100
LEARNING_RATE = 0.01
THRESHOLD = 0.25 #Ngưỡng dừng cho hàm

# Hàm Gradient
def gradient(LEARNING_RATE, NUMITERATION, THRESHOLD):
    # Khởi tạo nghiệm ban đầu (giá trị bắt đầu)
    w = np.array([0.,0.1,0.1]).reshape(-1,1)

    # Mảng chứa các giá trị của hàm loss
    Loss = np.zeros((NUMITERATION, 1))

    for i in range(0, NUMITERATION):
        # Tính giá trị dự đoán
        y_predict = sigmoid(np.dot(X, w))
        Loss[i] = -np.sum(np.multiply(y, np.log(y_predict)) + np.multiply(1-y, np.log(1-y_predict)))

        # gradient descent
        w = w - LEARNING_RATE * np.dot(X.T, y_predict-y)

        if Loss[i] <= THRESHOLD: break
    return w, Loss
w, Loss = gradient(LEARNING_RATE, NUMITERATION, THRESHOLD)
```

Cài đặt thuật toán

```
In [4]: plt.ylabel("Loss")
        plt.xlabel("Iteration")
        plt.plot(range(0, NUMITERATION), Loss, label="Value of The Loss function")
        plt.legend(loc="upper right")
        plt.grid()
        plt.show()
```



Cài đặt thuật toán

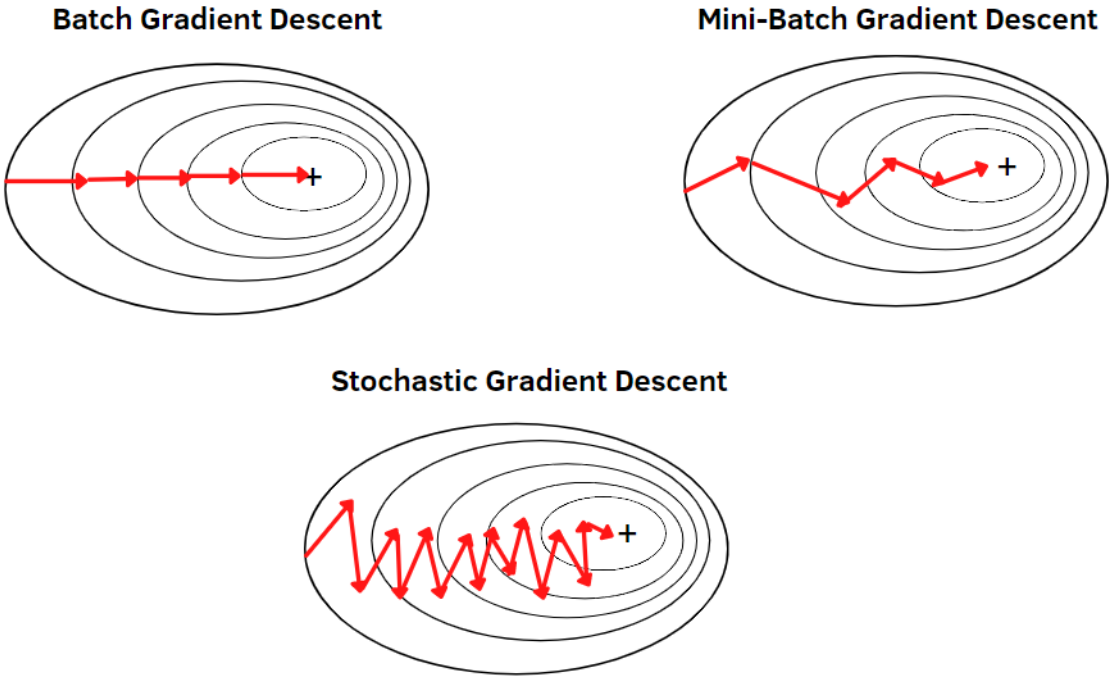
- **Xây dựng chương trình tìm các trọng số theo mô hình Hồi quy Logistic dựa trên thư viện Sklearn.**

In [5]:
```
# Logistic Regression dùng thư viện sklearn
from sklearn.linear_model import LogisticRegression


# Chuẩn bị dữ liệu huấn luyện
x1 = np.array([[6, 8, 9, 10, 9, 5, 8]]).T #n x 1
x2 = np.array([[2, 1, 3, 7, 9, 3, 12]]).T #n x 1

X = np.concatenate((x1,x2), axis = 1)
y = np.array([0, 0, 0, 1, 1, 0, 1]).T #n x 1


logis = LogisticRegression(random_state = 42)
logis.fit(X, y)

print( 'Coefficient : ', logis.coef_ )
print( 'Interception  : ', logis.intercept_ )
```

```
Coefficient :  [[0.33945107 0.8775603 ]]
Interception  :  [-7.72608038]
```

Stochastic Gradient descent

- SGD là một biến thể của GD
- GD sau mỗi một lần lặp sẽ cập nhật trọng số một lần còn SGD sẽ cập nhật n lần theo n điểm dữ liệu.
- Phù hợp với bài toán có nhiều điểm dữ liệu.



Bài tập về nhà

- Đọc lại toàn bộ kiến thức đã học trên lớp về mô hình hồi quy Logistic và phương pháp Gradient descent.
- Xây dựng mô hình hồi quy Logistic theo phương pháp tìm nghiệm của Gradient descent trên một bộ dữ liệu mẫu.
- Tìm hiểu và tính đạo hàm của hàm tanh
- Tìm hiểu về một số thư viện trong mô hình hồi quy Logistic.